Unicenter® CA-PLEU® Protection Log Extract Utility for Adabas

User Guide

4.1

Computer Associates®

Contents

Chapter 1: General Information

Chapter 2: Installing Unicenter CA-PLEU

Chapter 5: Command Reference Information

Chapter 6: SAVEFDTS Program

Chapter 7: User Exits

Chapter 1

General Information

This guide contains the information and instructions you need to install and use Computer Associates Unicenter CA-PLEU Protection Log Extract Utility for Adabas (Unicenter CA-PLEU). Unicenter CA-PLEU is a set of programs used with the Adabas database management system to extract information from the Adabas Protection Logs for processing by installation programs.

Purpose

This chapter provides a brief overview of materials and system requirements needed to install Computer Associates Unicenter CA-PLEU Protection Log Extract Utility for Adabas (Unicenter CA-PLEU).

Certain knowledge is required to complete the installation process and for the execution of Unicenter CA-PLEU. For example, you must be familiar with z/OS and OS/390 IEBCOPY or VSE/ESA LIBR utilities; Adabas Protection Logging; Adabas Format buffers and FDTs; and the Adabas ADARES PLCOPY and ADALOD compress and load utilities.

Installation Materials

Before beginning the installation procedure, make sure that you have the correct Unicenter CA-PLEU installation tape. Verify the volume serial number on the tape matches the volume serial number specified in the Product Maintenance Letter received with the product package. If you did not receive the proper tape, please contact the Computer Associates Order department at 1-800-841-8743.

In addition, verify that you received a Product Authorization letter containing a numeric password required for installation. If you did not receive the Product Authorization letter, please contact your local Technical Support group.

IBM users, make sure you also received a CA LMP Key Certificate for Unicenter CA-PLEU with the product package and installation tape. If you did not receive a Unicenter CA-PLEU LMP Key Certificate, contact your Computer Associates account manager or the Total License Care Hot Line at 1-800-338-6720.

System Requirements

The following system requirements are necessary for the successful installation and execution of Unicenter CA-PLEU:

- Adabas versions 7.1, 7.2, or 7.4
- z/OS and OS/390 operating systems that supports Adabas V7.1 or higher
- VSE/ESA operating systems that supports Adabas V7.1 or higher
- Fujitsu MSP operating systems that supports Adabas V7.1 or higher
- Hitachi VOS3 operating systems that supports Adabas V7.1 or higher

Release Dependencies

z/OS and OS/390 Operating Systems	All system functions are standard, and no release dependencies should be encountered.
VSE/ESA Operating Systems	Standard VSE/ESA function macros have been used, and no release dependencies should be encountered.
Fujitsu MSP Operating Systems	All system functions are standard, and no release dependencies should be encountered.
Hitachi VOS3 Operating Systems	All system functions are standard, and no release dependencies should be encountered.
Adabas	File definitions are read using the Adabas LF command, and no release dependencies should be encountered for this function.

Unicenter CA-PLEU processes Protection Logs from either Adabas version 7.1, 7.2, or 7.4. However, do not merge or concatenate Protection Logs from different versions and/or releases – that is, V6 and V7 Protection Logs or V7.1 and V7.4 Protection Logs.

CA Common Services for z/OS and OS/390

To help you quickly understand all that CA Common Services for z/OS and OS/390 offers, this section provides a brief description of the common services that can be used by Unicenter CA-PLEU.

The CA Common Services for z/OS and OS/390 are a group of system services that protect your investment in software products by helping you manage your data center more efficiently. The CA Common Services for z/OS and OS/390 offer individual benefits to the user.

The CA Common Services for z/OS and OS/390 that are used with, and benefit, Unicenter CA-PLEU is CAIRIM and CA LMP, which assist you in getting Unicenter CA-PLEU up and running.

Installing CA Common Services for z/OS and OS/390

According to your data center's specific needs, you can choose to use only certain CA Common Services for z/OS and OS/390. Some of the services are dependent on one or more other services to function properly when interfacing with your Computer Associates software. The interservice dependencies for the services used by Unicenter CA-PLEU are listed below.

If you are installing:	**You must also install:**
CAIRIM	No other services are required
CA LMP	CAIRIM

If the necessary services have not already been installed on your system, you must do so now. Refer to the *CA Common Services for z/OS and OS/390 Getting Started* for detailed instructions.

If the CA Common Services for z/OS and OS/390 services are not installed and you attempt to use Unicenter CA-PLEU, a S122 abend results.

CAIRIM

CAIRIM, CAI Resource Initialization Manager, is the common driver for a collection of dynamic initialization routines that eliminate the need for user SVCs, SMF exits, subsystems, and other installation requirements commonly encountered when installing systems software. These routines are grouped under the Computer Associates z/OS and OS/390 dynamic service code, S910. Some of the features of CAIRIM include:

Obtaining SMF data

- Verification of proper software installation
- Installation of z/OS and OS/390 interfaces
- Automatic startup of Computer Associates and other vendor products
- Proper timing and order of initialization
- No other services are required for proper operation.

Note: CAIRIM is mandatory for Unicenter CA-PLEU. It must be installed and started within 30 minutes of IPL time. CAIRIM is part of the CA Common Services for z/OS and OS/390.

CA LMP

The Computer Associates License Management Program (CA LMP) provides a standardized and automated approach to the tracking of licensed software. It uses common real-time enforcement software to validate the user's configuration. CA LMP reports on activities related to the license, usage, and financials of Computer Associates software solutions. The routines that accomplish this are integrated into the Computer Associates z/OS and OS/390 dynamic service code, S910 (the CAIRIM service). Some of the features of CA LMP include:

- Common Key Data Set can be shared among many CPUs
- "Check digits" are used to detect errors in transcribing key information
- Execution Keys can be entered without affecting any Computer Associates software solution that is already running
- There are no special maintenance requirements

Requirements

Unicenter CA-PLEU requires CA Common Services for z/OS and OS/390 at genlevel 9901 or above.

Refer to eSupport for additional Unicenter Services minimum genlevel requirements for your release of OS/390 or z/OS.

Using CA LMP

Unicenter CA-PLEU requires CA LMP (License Management Program), one of the Common Services, to initialize correctly. CA LMP also provides a standardized and automated approach to the tracking of licensed software.

CA LMP is provided as an integral part of CAIRIM (Resource Initialization Manager), another one of the Common Services. If CAIRIM has not already been installed on your system, you must do so now. Once CAIRIM has been installed or maintained at Service Level C1/9901 or higher, CA LMP support is available for all CA LMP—supported CA software solutions. See the *CA Common Services for z/OS and OS/390 Getting Started* guide for detailed instructions on installing CAIRIM.

Examine the CA LMP Key Certificate you received with your Unicenter CA-PLEU installation or maintenance tape. That certificate contains the following information:

Fields	**Descriptions**
Product Name	The trademarked or registered name of the CA software solution licensed for the designated site and CPUs.
Product Code	A two-character code that corresponds to Unicenter CA-PLEU.
Supplement	The reference number of your license for Unicenter CA-PLEU, in the format *nnnnnn - nnn*. This format differs slightly inside and outside North America, and in some cases may not be provided at all.
CPU ID	The code that identifies the specific CPU for which installation of Unicenter CA-PLEU is valid.
Execution Key	An encrypted code required by CA LMP for Unicenter CA-PLEU initialization. During installation, it is referred to as the LMP Code.
Expiration Date	The date *(ddmmmyy* as in 01AUG02) your license for Unicenter CA-PLEU expires.

Fields	Descriptions
Technical Contact	The name of the technical contact at your site responsible for the installation and maintenance of Unicenter CA-PLEU. This is the person to whom CA addresses all CA LMP correspondence.
MIS Director	The name of the Director of MIS, or the person who performs that function at your site. If the title but not the individual's name is indicated on the Certificate, you should supply the actual name when correcting and verifying the Certificate.
CPU Location	The address of the building where the CPU is installed.

The CA LMP execution key, provided on the Key Certificate, must be added to the CAIRIM parameters to ensure proper initialization of Unicenter CA-PLEU. To define a CA LMP execution key to the CAIRIM parameters, modify member KEYS in the OPTLIB data set.

The parameter structure for member KEYS is as follows:

```
PROD(pp) DATE(ddmmmyy) CPU(tttt-mmmm/ssssss) LMPCODE(kkkkkkkkkkkkkkkk)
```

Where:

pp—Required. The two-character product code. For any given CA LMP software solution, this code agrees with the product code already in use by the CAIRIM initialization parameters for earlier gen levels of that software solution.

The two-character product codes for Unicenter CA-PLEU are:

HS	z/OS and OS/390
HP	VSE/ESA

ddmmmyy—The CA LMP licensing agreement expiration date.

tttt-mmmm—Required. The CPU type and model (for example: 3090 - 600) on which the CA LMP software solution is to run. If the CPU type and/or model require less than four characters, blank spaces are inserted for the unused characters.

ssssss—Required. The serial number of the CPU on which the CA LMP software solution is to run.

kkkkkkkkkkkkkkkk—Required. The execution key needed to run the CA LMP software solution. This CA LMP execution key is provided on the Key Certificate shipped with each CA LMP software solution.

The following is an example of a control statement for the CA LMP execution software parameter. Although this example uses the Unicenter CA-PLEU two-character product code, the CA LMP execution key value is invalid and is provided as an example only!

```
PROD(HS) DATE(01AUG02) CPU(3090— —— —600 /370623) LMPCODE(52H2K06130Z7RZD6)
```

For a full description of the procedure for defining the CA LMP execution key to the CAIRIM parameters, see the *CA Common Services for z/OS and OS/390 Getting Started*.

Technical Support

If questions arise during the installation or operation of Unicenter CA-PLEU, or if you have suggestions regarding the use of the product, please call Computer Associates Technical Support:

- U.S. and Canadian customers: (425) 825-2770
- International customers: Contact your nearest Computer Associates office

Chapter 2 Installing Unicenter CA-PLEU

This chapter describes installation of Unicenter CA-PLEU Protection Log Extract Utility for Adabas (Unicenter CA-PLEU) for IBM z/OS, OS/390, and VSE/ESA systems; Fujitsu MSP; and Hitachi VOS3 systems.

- IBM z/OS, OS/390, and VSE/ESA users should proceed to the following section: Define the Unicenter CA-PLEU Computer Associates LMP Key.
- Fujitsu MSP and Hitachi VOS3 users should proceed directly to the section: Unicenter CA-PLEU Installation Worksheet on the following page.

Define the Unicenter CA-PLEU Computer Associates LMP Key

Note: This step is required for IBM z/OS, OS/390, and VSE/ESA users. The former Legent Product Code used in previous releases of Unicenter CA-PLEU is no longer supported.

Make sure you have received a CA LMP Key Certificate for Unicenter CA-PLEU with your product package and installation tape. If you did not receive a Unicenter CA-PLEU LMP Key Certificate, contact your Computer Associates account manager or the Total License Care Hot Line at (800) 338-6720.

You must define the CA LMP execution key to the CAIRIM parameters. Refer to the *CA Common Services for z/OS and OS/390 Getting Started* for a full description of the procedure.

Unicenter CA-PLEU Installation Worksheet

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

The JCL members you use in the installation steps that follow need to be modified to conform to your sites specifications. At a minimum, you must supply job card and data set name information in each JCL member.

To assist you in making the necessary JCL modifications, the following worksheet lists the variable names used in the distributed JCL members followed by a description of what the variable is used to identify.

Unicenter CA-PLEU Installation Worksheet

You may wish to record the values you assign to the environment variables for future reference on the lines provided.

Environment Variable	Description
ADADBID	Specify the value to be used for the Adabas DBID in the distributed JCL.
ADALOAD	Specify the value to be used for the Adabas load library in the distributed JCL.
ADANAME	Specify the value to be used for the Adabas name in the distributed JCL.
ADASIBA	Specify the value to be used for the Adabas Protection Logs in the distributed JCL.
ADASVC	Specify the value to be used for the Adabas SVC in the distributed JCL.
ADAVER	Specify the value to be used for the ADAVER GLOBALS parameter in the distributed JCL and identifies the version of Adabas used to create the Protection Logs.

Environment Variable	Description
`DISKUNIT`	Specify the value to be used for the generic DASD unit name in the distributed JCL.
`HLQ`	Specify the value to be used for the high-level qualifier of data sets in the distributed JCL.
`JOBCARD1`	Specify the value to be used for jobcard 1 in the distributed JCL.
`JOBCARD2`	Specify the value to be used for jobcard 2 in the distributed JCL.
`MLQ`	Specify the value to be used for the mid-level qualifier of data sets in the distributed JCL.
`PLEUPSWD`	Specify the eight-digit number found in the Product Authorization letter included in the product package.
`PRODVOL`	Specify the value to be used for the DASD volume on which the product libraries are installed.
`TAPEUNIT`	Specify the value to be used for the generic TAPE unit name in the distributed JCL used to unload product libraries.
`TAPEVOL`	Specify the value to be used for the VOLSER of the distribution tape in JCL used to unload product libraries.

IBM z/OS and OS/390 Users

IBM z/OS and OS/390 users proceed with the installation steps beginning with Installation Instructions for z/OS and OS/390 in this chapter.

IBM VSE/ESA Users

IBM VSE/ESA users should proceed with the installation steps beginning with Installation Instructions for VSE/ESA Systems in this chapter.

Fujitsu MSP Users

Fujitsu MSP users should proceed with the installation steps beginning with Installation Instructions for Fujitsu MSP Systems in this chapter.

Hitachi VOS3 Users

Hitachi VOS3 users should proceed with the installation steps beginning with Installation Instructions for Hitachi VOS3 Systems in this chapter.

Installation Instructions for z/OS and OS/390

This section describes the steps necessary to install Unicenter CA-PLEU on IBM z/OS and OS/390 systems.

Step 1: Create and Submit Initial Distribution Tape Extract Job

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

In this step you create a JCL job stream to copy the Unicenter CA-PLEU INSTALL library from the Unicenter CA-PLEU distribution tape.

The names of the Unicenter CA-PLEU libraries on the tape are different from those of earlier versions. This is intended to help avoid mixing components from different versions.

Note: Do **not** merge libraries from this tape with libraries from previous releases.

The distribution tape VOLSER is:

ASP*vvv*

Where *vvv* is the version, release, and sm level for the current release. Be sure to check the tape label for the correct VOLSER.

Use the following steps to create the INSTALL library:

1. Create the following JCL:

```
//PLEUINST  JOB (acct info),'COPY INSTALL LIB',CLASS=x
//PLEUSTP1 EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT3   DD UNIT=diskunit,SPACE=(TRK,(1,1))
//SYSUT4   DD UNIT=diskunit,SPACE=(TRK,(1,1))
//TAPE     DD DSN=CA.PLEUVvvv.INSTALL,DISP=(OLD,KEEP),
//            UNIT=tapeunit,VOL=SER=ASPvvv,
//            LABEL=(19,SL)
//DISK     DD DSN=hlq.mlq.PLEUVvvv.INSTALL,
//            DISP=(NEW,CATLG,DELETE),VOL=SER=prodvol,
//            UNIT=diskunit,SPACE=(TRK,(5,2,5))
//SYSIN    DD *
COPY INDD=TAPE,OUTDD=DISK
/*
//
```

2. Provide the necessary site-dependent information in the JOB card and values for those parameters specified in italicized letters.

3. Submit the job. If you receive a return code other than zero, review the cause of the errors, take the necessary corrective action, and resubmit the job.

Step 2: Read PLREADME

Read the PLREADME member in the Unicenter CA-PLEU INSTALL library for additional information you may need for this installation and/or product use.

Step 3: Modify the Unicenter CA-PLEU Installation Edit Macro

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

We recommend that you use the edit macro PLEDIT in the INSTALL library to quickly and accurately make changes to the PDS members used to install Unicenter CA-PLEU. You modify PLEDIT with the values entered on the Unicenter CA-PLEU Installation Worksheet.

Use the following steps to modify PLEDIT:

1. Replace the right most parameter of each ISREDIT CHANGE macro statement with the corresponding values entered on the Unicenter CA-PLEU Installation Worksheet.

2. Store PLEDIT in a library that is concatenated to the SYSPROC DD in your TSO logon procedure.

3. Each time you edit an installation member, type PLEDIT on the TSO COMMAND line and press Enter to replace the variable names in the distributed JCL with your specifications.

Step 4: Allocate and Load Product Libraries

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

Use the following steps to allocate and unload the Unicenter CA-PLEU product libraries from the distribution tape:

1. Modify PLUNLOAD.

   Use the PLEDIT macro to change the values of the environment variables in the PLUNLOAD member in the Unicenter CA-PLEU INSTALL library as specified on the Unicenter CA-PLEU Installation Worksheet.

2. Submit the PLUNLOAD JCL.

   If you receive a non-zero return code, correct the JCL and resubmit the job. Use PLDELETE in the INSTALL library to delete any of the data sets allocated with PLUNLOAD.

Installation is now complete.

Installation Instructions for VSE/ESA Systems

The installation of Unicenter CA-PLEU consists of a LIBR RESTORE to copy the Unicenter CA-PLEU sublibrary from the distribution tape to the appropriate sublibrary on your system. This section describes the steps necessary to install Unicenter CA-PLEU on IBM VSE/ESA systems.

Step 1: Create and Submit the LIBR Restore Job

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

The name of the Unicenter CA-PLEU data set on this tape is different from earlier versions. This is intended to help avoid mixing different versions.

Note: Do **not** merge the sublibrary from this tape with sublibraries from previous releases.

The distribution tape VOLSER is:

ASP*vvv*

Where *vvv* is the version, release, and sm level of the current release. Be sure to check the tape label for the correct VOLSER.

Use the following steps to create the Unicenter CA-PLEU sublibrary:

1. Create the following LIBR JCL to restores the Unicenter CA-PLEU sublibrary to the library.sublibrary appropriate for your site. The Unicenter CA-PLEU sublibrary requires less than 250 library blocks.

```
* $$ JOB JNM=PLEULIBR,CLASS=x
* $$ LST CLASS=x
// JOB LIBRRSTR
// OPTION LOG
// ASSGN SYS006,cuu
// TLBL  PLBKUP,'CA.PLEU.Vvvv',0,ASPvvv
// MTC REW,SYS006
// MTC FSF,n
// EXEC LIBR,SIZE=256K
   DEFINE   S=yourlib.PLEUvvv
   RESTORE   S=CADBADB.PLEU : yourlib.PLEUvvv  T=SYS006  -
             TL=PLBKUP  R=YES
/*
/&
* $$ EOJ
```

2. Provide the necessary site-dependent information in the POWER cards and values for those parameters specified in italicized letters.

3. Submit the job. If you receive a return code other than zero, review the cause of the errors, take the necessary corrective action, and resubmit the job.

Step 2: Read @README

Read the @README.V member in the Unicenter CA-PLEU sublibrary created in Step 1 for additional information you may need for this installation and/or product use.

Step 3: Update the System History File

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

In this step you create new Unicenter CA-PLEU product and component entries in the System History file.

Use the following steps to modify the MSHPARCJ.V JCL member and update the System History file:

1. Punch MSHPARCJ.V from the Unicenter CA-PLEU sublibrary to your editor.
2. Modify MSHPARCJ to meet your installation standards.
3. Submit MSHPARCJ.
4. Check the output for messages indicating the successful completion of the job.

Installation is now complete.

Installation Instructions for Fujitsu MSP Systems

This section describes the steps necessary to install Unicenter CA-PLEU on Fujitsu MSP systems.

Step 1. Create and Submit Initial Distribution Tape Extract Job

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

In this step you create a JCL job stream to copy the Unicenter CA-PLEU INSTALL library from the Unicenter CA-PLEU distribution tape.

The names of the Unicenter CA-PLEU libraries on the tape are different from those of earlier versions. This is intended to help avoid mixing components from different versions.

Note: Do **not** merge libraries from this tape with libraries from previous releases.

The VOLSER of the distribution tape is

ASPvvv

Where *vvv* is the current version, release, and sm level for the current release. Be sure to check the tape label for the correct VOLSER.

Use the following steps to create the INSTALL library in Fujitsu MSP environments:

1. Create the following JCL:

```
//PLEUINST  JOB (acct info),'COPY INSTALL LIB',CLASS=x
//PLEUSTP1 EXEC PGM=JSECOPY
//SYSPRINT DD SYSOUT=*
//SYSUT3   DD UNIT=diskunit,SPACE=(TRK,(1,1))
//SYSUT4   DD UNIT=diskunit,SPACE=(TRK,(1,1))
//TAPE     DD DSN=CA.PLEUVvvv.INSTALL,DISP=(OLD,KEEP),
//            UNIT=tapeunit,VOL=SER=ASPvvv,
//            LABEL=(19,SL)
//DISK     DD DSN=hlq.mlq.PLEUVvvv.INSTALL,
//            DISP=(NEW,CATLG,DELETE),VOL=SER=prodvol,
//            UNIT=diskunit,SPACE=(TRK,(5,2,5))
//SYSIN    DD *
COPY INDD=TAPE,OUTDD=DISK
/*
//
```

2. Provide the necessary site-dependent information in the JOB card and values for those parameters specified in italicized letters.

3. Submit the job. If you receive a return code other than zero, review the cause of the errors, take the necessary corrective action, and resubmit the job.

Step 2: Read PLREADME

Read the PLREADME member in the Unicenter CA-PLEU INSTALL library for additional information you may need for this installation and/or product use.

Step 3: Modify the Unicenter CA-PLEU Installation Edit Macro

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

We recommend that you use the edit macro PLEDIT in the INSTALL library to quickly and accurately make changes to the PDS members used to install Unicenter CA-PLEU. You modify PLEDIT with the values entered on the Unicenter CA-PLEU Installation Worksheet.

Use the following steps to modify PLEDIT:

1. Replace the right most parameter of each ISREDIT CHANGE macro statement with the corresponding values entered on the Unicenter CA-PLEU Installation Worksheet.
2. Store PLEDIT in a library that is concatenated to the SYSPROC DD in your TSO logon procedure.
3. Each time you edit an installation member, type PLEDIT on the TSO COMMAND line and press Enter to replace the variable names in the distributed JCL with your specifications.

Step 4: Allocate and Load Product Libraries

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

Use the following steps to allocate and unload the Unicenter CA-PLEU product libraries from the distribution tape:

1. Modify PLUNLD$F.

   Use the PLEDIT macro to change the values of the environment variables in the PLUNLD$F member in the Unicenter CA-PLEU INSTALL library as specified on the Unicenter CA-PLEU Installation Worksheet.

2. Submit the PLUNLD$F JCL.

   If you receive a non-zero return code, correct the JCL and resubmit the job. Use PLDELETE in the Unicenter CA-PLEU INSTALL library to delete any data sets allocated with PLUNLD$F.

Installation is now complete.

Installation Instructions for Hitachi VOS3 Systems

This section describes the steps necessary to install Unicenter CA-PLEU on Hitachi VOS3 systems.

Step 1. Create and Submit Initial Distribution Tape Extract Job

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

In this step you create a JCL job stream to copy the Unicenter CA-PLEU INSTALL library from the Unicenter CA-PLEU distribution tape.

The names of the Unicenter CA-PLEU libraries on the tape are different from those of earlier versions. This is intended to help avoid mixing components from different versions.

Note: Do **not** merge libraries from this tape with libraries from previous releases.

The VOLSER of the distribution tape is

ASPvvv

where *vvv* is the current version, release, and sm level for the current release. Be sure to check the tape label for the correct VOLSER.

Use the following steps to create the INSTALL library in Hitachi VOS3 environments:

1. Create the following JCL:

```
//PLEUINST  JOB (acct info),'COPY INSTALL LIB',CLASS=x
//PLEUSTP1 EXEC PGM=JSDPCPY
//SYSPRINT DD SYSOUT=*
//SYSUT3   DD UNIT=diskunit,SPACE=(TRK,(1,1))
//SYSUT4   DD UNIT=diskunit,SPACE=(TRK,(1,1))
//TAPE     DD DSN=CA.PLEUVvvv.INSTALL,DISP=(OLD,KEEP),
//            UNIT=tapeunit,VOL=SER=ASPvvv,
//            LABEL=(17,SL)
//DISK     DD DSN=hlq.mlq.PLEUVvvv.INSTALL,
//            DISP=(NEW,CATLG,DELETE),VOL=SER=prodvol,
//            UNIT=diskunit,SPACE=(TRK,(15,2,15))
//SYSIN    DD *
COPY INDD=TAPE,OUTDD=DISK
/*
//
```

2. Provide the necessary site-dependent information in the JOB card and values for those parameters specified in italicized letters.

3. Submit the job. If you receive a return code other than zero, review the cause of the errors, take the necessary corrective action, and resubmit the job.

Step 2: Read PLREADME

Read the PLREADME member in the Unicenter CA-PLEU INSTALL library for additional information you may need for this installation and/or product use.

Step 3: Modify the Unicenter CA-PLEU Installation Edit Macro

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

We recommend that you use the edit macro PLEDIT in the INSTALL library to quickly and accurately make changes to the PDS members used to install Unicenter CA-PLEU. You modify PLEDIT with the values entered on the Unicenter CA-PLEU Installation Worksheet.

Use the following steps to modify PLEDIT:

1. Replace the right most parameter of each ISREDIT CHANGE macro statement with the corresponding values entered on the Unicenter CA-PLEU Installation Worksheet.
2. Store PLEDIT in a library that is concatenated to the SYSPROC DD in your TSO logon procedure.
3. Each time you edit an installation member, type PLEDIT on the TSO COMMAND line and press Enter to replace the variable names in the distributed JCL with your specifications.

Step 4. Allocate and Load Product Libraries

Note: This step is required whether you are installing for the first time or migrating from a previous release of Unicenter CA-PLEU.

Use the following steps to allocate and unload the Unicenter CA-PLEU product libraries from the distribution tape:

1. Modify PLUNLD$H.

   Use the PLEDIT macro to change the values of the environment variables in the PLUNLD$H member in the Unicenter CA-PLEU INSTALL library as specified on the Unicenter CA-PLEU Installation Worksheet.

2. Submit the PLUNLD$H JCL.

   If you receive a non-zero return code, correct the JCL and resubmit the job. Use PLDELETE in the Unicenter CA-PLEU INSTALL library to delete any data sets allocated with PLUNLD$H.

3. Modify PLLKED$H.

   Use the PLEDIT macro to change the values of the environment variables in the PLLKED$H member in the Unicenter CA-PLEU INSTALL library as specified on the Unicenter CA-PLEU Installation Worksheet.

   Note: The BLKSIZE in the DCB cards can be modified to best suit your DASD environment.

4. Submit the PLLKED$H JCL.

   If you receive a non-zero return code, correct the JCL and resubmit the job.

Installation is now complete.

Chapter

3

Unicenter CA-PLEU Overview

This chapter provides general information about the Unicenter CA-PLEU Protection Log Extract Utility for Adabas (Unicenter CA-PLEU). Available functions are summarized and possible applications are suggested.

Unicenter CA-PLEU Features

The Unicenter CA-PLEU utility is used with the Adabas database management system to extract information from Adabas Protection Logs for processing by installation programs and has the following features:

- Reads a sequential (not dual or multiple) database Protection Log, which may be a tape or disk file.
- Processes Protection Logs created by Adabas version 7.1, 7.2, or 7.4.
- Only one pass of the Protection Log is necessary to produce an unlimited number of output extract files, each with its own format.
- The data extracted from the Protection Log can be written in any of three formats:
  - A fixed format specified by a standard Adabas format buffer
  - A format suitable for input to the Adabas compress and load utilities
  - A format which can be used to update an audit trail database through direct calls to Adabas
- Each extract can select records from the Protection Log based on Adabas file number, date, time, terminal-ID, user-ID, ISN (internal sequence number), TSN (transaction sequence number), Adabas command, and before and/or after images.
- Default output file characteristics are selected to optimize performance.
- It can use current field definitions stored within the Adabas database or field definitions saved in a sequential file at the time the Protection Log was written.
- Unicenter CA-PLEU can read the Protection Log tape backwards, extracting data in the reverse order of its creation; this feature can be useful when trying to find the last access to a particular ISN or when trying to back off a set of updates.

Applications

The output of Unicenter CA-PLEU has a variety of potential applications:

- Audit trails: detail and summary reporting to satisfy auditors' requirements to know:
  - Who is modifying data
  - What modifications have been made in summary and detail
  - That balance totals remain consistent
- Development validation: verification that expected data changes are taking place and production of summary totals validate batch updates against a file.
- Application error recovery: discovery of causes of incorrect data and application-dependent recovery of invalid or incorrectly entered updates.

Data Flow

Unicenter CA-PLEU reads the Adabas sequential (not dual or multiple) Protection Log and creates formatted output data sets containing before and after images of records selected from the log. The format of these output records and criteria used for selecting records are specified by the user through card images read by Unicenter CA-PLEU. There are three types of command statements.

- GLOBALS statements are used to define the processing environment for Unicenter CA-PLEU; it is optional and only one GLOBALS statement may be specified.
- SELECT statements are used to specify record selection criteria that apply to the entire execution of Unicenter CA-PLEU; it is optional and only one may be specified.
- EXTRACT statements are used to specify record selection criteria for individual Adabas files and output data sets (called extracts). In addition, EXTRACT statements specify the format of the records to be written into the output data set. At least one EXTRACT statement is required for each execution of Unicenter CA-PLEU, and there may be multiple EXTRACT statements.

Each EXTRACT statement provides the output format for a single Adabas file for a single output data set. The records for a single Adabas file can be extracted in several formats and written to several output data sets in a single execution of Unicenter CA-PLEU. In addition, extracts for multiple Adabas files can be created in a single execution of Unicenter CA-PLEU.

An additional output data set containing standard printer output describes the execution of Unicenter CA-PLEU.

Features

One Pass Operation

Unicenter CA-PLEU supports a wide variety of potential applications of the Adabas Protection Log and makes efficient use of machine resources. Some Adabas applications update several Adabas files in the course of their work. An audit trail for such an application should contain all updates in a single data set, regardless of which Adabas files were altered. To satisfy this requirement, Unicenter CA-PLEU can extract updates for multiple Adabas files in one pass of the Protection Log and place the extracted data into a single output data set.

Multiple Output Formats

Unicenter CA-PLEU provides support for Adabas files updated by more than one application; each such application may have unique auditing requirements. It is desirable for each application to be able to define its own extract format so it can get necessary data from the Protection Log. Unicenter CA-PLEU can extract updates to individual Adabas files in several formats, writing each format to its own data set, and it can do so in a single pass of the Protection Log.

These features permit an installation to satisfy many requirements for data from the Protection Log in a single pass of the Protection Log tape. Each execution of Unicenter CA-PLEU reads the Protection Log once; carefully planned, that single execution can provide each application with a formatted file that satisfies its unique auditing requirements. Moreover, as auditing requirements change or new applications are added, the new requirements can be satisfied without affecting other audit trails.

Multiple Adabas version Support

Unicenter CA-PLEU can process Protection Logs created Adabas version 7.1, 7.2, or 7.4.

Warning: Do NOT merge or concatenate Protection Logs from different versions and/or releases – that is, V6 and V7 Protection Logs or V7.1 and V7.4 Protection Logs.

Note: This version of Unicenter CA-PLEU does not support Adabas version 6 or earlier.

ADASEL Compatible

Some installations may wish to copy data from the Protection Log into an Adabas file so it is available for online processing. Unicenter CA-PLEU allows Protection Log records to be extracted in a format similar to that produced by the Software AG utility ADASEL. These extracts can then be loaded to an Adabas file using the standard Adabas compress and load utilities.

Post-Processing Option

To provide flexibility, a user exit can be specified for each extract file that enables the installation to perform additional processing on the Protection Log records before they are written to the extract file. This processing can result in the insertion of additional records, the deletion of records, or the modification of records before they are written out.

An additional output format is provided that generates Protection Log records in a format that contains only non-null fields, along with a corresponding format buffer. This record and format buffer may be processed by the user exit to add records to an audit database.

Protects Against Field Changes

A major feature of Adabas is its ability to support changes to the field definitions of a file without affecting usability of the existing data records. Unicenter CA-PLEU supports this flexibility by providing a means for saving field definitions so that old Protection Logs can be processed against the field definitions in effect at the time of their creation. This feature also allows Protection Log processing to take place without Adabas being active.

Protection Log Considerations

Differing Adabas versions

Unicenter CA-PLEU reads input data from the sequential file, but not dual or multiple, Protection Log produced by Adabas. This log contains a number of record formats documented in the *Adabas Internals* class student guide. The extract program processes the record types corresponding to before and after record images (data) and those for ET, C1, C3, and C5 commands.

Do NOT merge or concatenate Protection Logs from different versions and/or releases—that is, V6 and V7 Protection Logs or V7.1 and V7.4 Protection Logs.

Mass Updates

Remember the Protection Log does not contain before/after images for the mass updates that are made by Adabas utilities. For example, the loading of an Adabas file is not reflected in the Protection Log. Installations need to make their own provisions for auditing these activities. This can generally be done using manual techniques because the utilities are usually run by a central group of database administrators, whose actions can be closely monitored. For an installation's audit trail to be complete, procedures for documenting mass updates must be defined and followed.

The Log Tape

While logging can occur either to disk or to tape, most installations copy the logs to tape on a regular basis. Therefore, this document generally refers to the Adabas Protection Log input to this utility as the “log tape.”

Full Protection Logs

Management of the Protection Log is documented in the *Adabas DBA Reference Manual*. When dual or multiple protection logging is used, it is critical to the audit trail that there be no failure of ADARES/COPY jobs when the disk logs become full. Any such failure not only results in holes in the audit trail, but also prevents recovery from a disk failure on the database packs.

Organizing Log Tapes

When records are being extracted within a user-specified time interval, the amount of processing is reduced when only those tape volumes which contain records generated within that time interval are searched. Thus, the organization of the log tapes can have a significant effect on the amount of resources required to extract data from the log tapes. A record of execution of ADARES/PLCOPY and the volumes used are useful in this regard.

Field Definition Considerations

Default Definitions

Unicenter CA-PLEU uses standard Adabas field definitions to process the data in the Protection Log. By default, it uses the field definitions in the Adabas database identified in standard ADARUN cards.

Using SAVEFDTS

Because Unicenter CA-PLEU reads only those field definitions for the Adabas files requested and because it only reads each field definition once, the effect on Adabas performance is negligible. However, there are valid reasons why you may not want to use the active field definitions:

- To process a Protection Log where one or more field definitions have been changed since it was created
- To process Protection Log records when Adabas is not active
- To process a Protection Log on a different machine than the one on which it was created

All of these situations can be satisfied by using the program SAVEFDTS to copy the field definitions to a sequential file. See the chapter "SAVEFDTS Program" for more information.

The field definitions required by Unicenter CA-PLEU can then be read from one or more sequential files, rather than from the Adabas database. By managing the SAVEFDTS output files and specifying the correct information on Unicenter CA-PLEU control cards, the appropriate field definitions for each Adabas file on the Protection Log can be used.

Unicenter CA-PLEU Output Files

Output File Record Header

The following DSECT maps the Unicenter CA-PLEU header portion of the output records for Adabas version 7 Protection Logs. Every such record has the same 128-byte header followed by the data taken from the Protection Log record.

- Depending on what Adabas writes into the Protection Log record, the contents of the OUT5NODE field may or may not be the terminal-ID. The terminal-ID may be included as part of the OUT5UNID field.
- The contents of the OUT5UNID field are reported directly from the Protection Log rather than from a Unicenter CA-PLEU GLOBALS keyword.

The format, content, and meaning of the sub-fields of OUT5UNIQ are dependent on the operating system and TP monitor being used. Users should contact Software AG for an explanation of the contents of these sub-fields.

Record Header DSECT

The Unicenter CA-PLEU record header DSECT is shown below.

```
  OFFSET
Hex    Dec        FIELD            FORM       DESCRIPTION
--------------------------------------------------------------------------------
0000   0000       OUT5BUFF DSECT
0000   0000       OUT5RECL DS      H          Record length
0002   0002       OUT5ZERO DS      H          Zeroes for RDW
0004   0004       OUT5USER DS      CL8        User-ID or time stamp
000C   0012       OUT5NODE DS      CL4        Node-ID
                  *                           Field in Adabas ACB for
                  *                           this user's first Adabas Command
0010   0016       OUT5FILE DS      H          Adabas file number
0012   0018       OUT5TYPE DS      CL2        Image type
                  *                CL2'A'     After image
                  *                CL2'B'     Before image
                  *                CL2'C1'    C1 record
                  *                CL2'C3'    C3 record
                  *                CL2'C5'    C5 record
                  *                CL2'ET'    ET record
0014   0020       OUT5ISN  DS      F          ISN from data field
0018   0024       OUT5DATE DS      CL8        YYYYMMDD of PLOG record
0020   0032       OUT5TIME DS      CL6        HHMMSS of PLOG record
0026   0038       OUT5DBNM DS      CL8        DBNAME from GLOBALS
002E   0046       OUT5DBID DS      H          DBID
0030   0048       OUT5CPID DS      CL8        CPU-ID from GLOBALS
0038   0056       OUT5DATL DS      H          Length of OUT5DATA area in bytes
003A   0058       * 2 Slack Bytes are created by the Full word alignment of TSN
003C   0060       OUT5TSN  DS      F          Transaction sequence number
0040   0064       OUT5CPNM DS      CL4        Check point name
0044   0068       OUT5RABN DS      F          RABN
0048   0072       OUT5CLUS DS      H          Cluster number (maybe)
004A   0074       OUT5PRTY DS      H          Priority
004C   0076       OUT5CMDC DS      F          Command count
0050   0080       OUT5BLCT DS      F          Block count from PLOG tape
0054   0084       OUT5RCND DS      H          Record number within block
0056   0086       OUT5SESS DS      H          Adabas session number
0058   0088       OUT5UNIQ DS      0CL28      General unique identifier
0058   0088       OUT5UNCP DS      XL8        General unique identifier, CPU-ID.
0060   0096       OUT5UNVM DS      CL8        General unique identifier, VM-ID.
0068   0104       OUT5UNOS DS      XL4        General unique identifier, OS-ID.
006C   0108       OUT5UNID DS      XL8        General unique identifier, USER-ID.
0074   0116       OUT5WRBN DS      XL2        Work-RABN-chain
0076   0118                DS      XL10       Reserved
0080   0128       OUT5HDRL EQU     *-OUT5BUFF Header length
0080   0128       OUT5DATA EQU     *          Start of output data.
```

Note: Beginning with V3.4, the output file record header field OUT5DATE has been expanded to 8 bytes and now includes the century. The overall length of the header is still 128 bytes but the offsets of individual fields have changed.

Output Formats

One out of three output formats must be requested:

A fixed format that produces records that can be processed by any business-oriented language, for example: COBOL, PL/I, Natural, or report writer.

Fixed format records contain only those fields requested by the user. The fields to be included and the format of each output field are specified using a standard Adabas command-level format buffer specification with certain restrictions (see Adabas Format Specifications).

- A format similar to that produced by ADASEL, the Adabas utility, which processes Protection Log records for individual files.

  For Adabas version 7 Protection Logs, the Unicenter CA-PLEU record header precedes the decompressed data. This header contains all of the information that the ADASEL LOGINFO header does, plus additional items. Note, however, that the sequence of items in the Unicenter CA-PLEU header is different from that of the ADASEL header.

- Two formats, ADDFORM and ADDFORM-LONG contain only non-null fields and are accompanied by a format buffer built by Unicenter CA-PLEU. These formats result in output records whose length and contents are not predictable because they are data-dependent. Unicenter CA-PLEU creates a format buffer for every output record and makes it available to the user exit. The format buffer can then be used, either by Adabas or by a user's own routine, to identify the individual data items available in the output record.

Sequence of Output Records

Output of Unicenter CA-PLEU is in the sequence of input from the log tape. Therefore, the output is in chronological sequence only if the tapes are requested in chronological sequence and the tape is not being read backwards.

z/OS and OS/390 users should note that a generic mount of a generation data set (generations not specified) results in mounting the tapes in the reverse of chronological order.

Note: In cases where the log tape is not being read backwards, the output from Unicenter CA-PLEU reflects the log tape in that before images precede after images. A before image followed by an after image for the same ISN for a file reflects a database UPDATE. It is possible for the before and after images to have different time-stamps. This happens because the time-stamp indicates the time the Protection Log block was written, so if the before image and after image are in different blocks their time-stamps probably differ. Also, since the user exit can delete and insert records, it is possible for the user exit to alter the order of the records from what has been described here.

Output File Limitations

Unicenter CA-PLEU imposes no limit on the number of extract requests. Except for VSE/ESA, Unicenter CA-PLEU imposes no limit on the number of output files. The maximum number of output files in VSE/ESA, while not unlimited, is felt to be large enough to meet all user needs. VSE/ESA users can create up to 10 tape files and 20 disk files simultaneously.

For all operating systems, practical limitations are based upon availability of devices and region/partition size.

Output Devices

The user may request that extracted before and after images from several Adabas files be written to the same output data set; this is desirable if an application updates data in more than one Adabas file. This allows the application's entire audit trail to be contained within a single data set.

The user may also request that extracted images for a single Adabas file be written to more than one data set; this is desirable if an Adabas file was updated by more than one application. The multiple extracts can each have their own format, so each extract can be tailored to its intended use.

The user should have some idea of how many output records will be generated from each extract. It is suggested that large output data sets be directed to available tape drives, and smaller data sets be allocated to disk.

Output Block Sizes

If no block sizes are specified in the JCL or on the control cards in VSE/ESA, output block sizes are selected on the basis of the device.

In z/OS and OS/390, the blocking defaults to 32760 for disk and tape devices. When using devices with track sizes larger than 32760, the default blocksize is 32760. This may result in inefficient use of the disk space, so a smaller blocksize may be appropriate.

In VSE/ESA, defaults are taken depending on whether the device is disk or tape. These block sizes are fairly large, and thus require large buffers to be allocated. If a large number of output data sets are being generated, the user may wish to specify block sizes smaller than the defaults.

Region/Partition Size

Virtual storage is used for field definition and other tables. The size of this area is dependent on the number of files processed and the number of extract requests generated. The size of these tables is not ordinarily a limiting factor in a system with virtual storage.

Chapter 4

Executing Unicenter CA-PLEU

Unicenter CA-PLEU Protection Log Extract Utility for Adabas (Unicenter CA-PLEU) operates in the same fashion as any other batch Adabas program. Its only access to the Adabas nucleus is to issue LF commands to read field definitions if the active Adabas field definitions are being used. If field definitions are being read from sequential files created by SAVEFDTS, no access is made to the Adabas nucleus, and Adabas does not have to be active.

Input Data Sets

There are four input data sets:

- The control card input stream containing the extract requests. This is standard card-image input.
  - z/OS and OS/390, MSP, and VOS3 ddname: SYSIN
  - VSE/ESA unit: SYSIPT
- The log tape(s) to be processed. These are the sequential nucleus Protection Log output, or the ADARES/PLCOPY output from the dual or multiple Protection Logs (not the dual or multiple disk files themselves).
  - z/OS and OS/390, MSP, and VOS3 ddname: SIBA
  - VSE/ESA unit: SYS010, filename SYS010
- The Adabas ADARUN control cards for execution of batch jobs at your installation.
  - z/OS and OS/390, MSP, and VOS3 ddname: DDCARD
  - VSE/ESA unit: SYS000, file name CARD

  This may be assigned to SYSIPT if the control cards follow the utility control cards with an intervening end-of-file indicator. This data set is not required if all field definitions are being read from data sets created by SAVEFDTS.
- The sequential data sets created by SAVEFDTS that contain field definitions.
  - z/OS and OS/390, MSP, and VOS3 ddnames: Specified by the user on the GLOBALS statement
  - VSE/ESA unit: From SYS041 to SYS050, file names: SYS041 to SYS050

Output Data Sets

There are a variable number of output data sets:

- Standard print output, containing error messages and execution statistics.
  - z/OS and OS/390, MSP, and VOS3 ddname: SYSPRINT
  - VSE/ESA unit: SYSLST
- One output data set for each output file name specified in the extract requests. These data sets are always formatted as variable-blocked. The blocksizes are selected depending on their device types. The blocksize may be overridden by specifying it in the JCL or on the VSE/ESA control card.
  - z/OS and OS/390, MSP, and VOS3 ddnames: As specified on the EXTRACT control cards
  - VSE/ESA units: From SYS011 to SYS040, file names: SYS011 to SYS040

Execution in z/OS and OS/390, MSP, and VOS3

Sample JCL

Unicenter CA-PLEU is executed as a standard job or job step. Following is a sample job control deck for z/OS and OS/390, MSP, and VOS3 together with an explanation of each DD card. (See the sample JCL member PLEUEXTJ in the SOURCE library.)

```
//JOBCARD1
//JOBCARD2
//STEP1     EXEC PGM=DBGADEXT
//STEPLIB   DD DSN=CA.PLEUVvvv.LOAD,DISP=SHR
//          DD DSN=ADABAS.LOADLIB,DISP=SHR
//DDCARD    DD *
ADARUN   MODE=MULTI,DBID=xxxx,SVC=nnn (,...)
//SYSPRINT DD SYSOUT=A
//FDTSEXT1 DD DSN=SAVEFDTS.OUTPUT1,DISP=SHR
//FDTSEXT3 DD DSN=SAVEFDTS.OUTPUT3,DISP=SHR
//SIBA     DD DSN=ADABAS.DATABASE.SIBA,DISP=OLD
//FILEEXT1 DD DSN=EXTRACT1.OUTPUT,UNIT=SYSDA,
//            DISP=(NEW,CATLG),SPACE=(...)
//FILEEXT2 DD DSN=EXTRACT2.OUTPUT,
//            UNIT=TAPE,DISP=(NEW,CATLG)
//FILEEXT3 DD DSN=EXTRACT3.OUTPUT,
//            UNIT=TAPE,DISP=(NEW,CATLG),
//            DCB=BLKSIZE=4800
.
.
.
//FILEEXTN DD...
//SYSIN DD *
GLOBALS FDTDDNAME=((3,FDTSEXT1),(13,FDTSEXT3))
        ADAVER=71
        ;
        SELECT  STARTDATE=19990420
                 ENDDATE=19990420
                STARTTIME=080000
```

```
                 ENDTIME=170000;
        EXTRACT FILE=3 DDNAME=FILEEXT1
                EXITNAME=EXIT003
                FORMAT='AA,BB,...'
                TERMID=TRM1,TRM2,... ;
        EXTRACT FILE=5 DDNAME=FILEEXT2
                FORMAT='AA,BB,...'
                USERID=USER1,USER2,...;
        EXTRACT FILE=13 DDNAME=FILEEXT3
                FORMAT='AA,BB,...' ;
        .
        .
        .
        EXTRACT FILE=NN DDNAME=FILEEXTN
                FORMAT='AA,BB,...'
                STARTTIME=090000 ENDTIME=120000;
/*
```

Job Control Language Explanation

The following explanations correspond to the z/OS and OS/390, MSP, and VOS3 control cards above.

- //STEPLIB is required. It identifies the load library which contains the DBGADEXT program. This or a concatenated data set must contain the load module EXIT003 requested in the EXTRACT statement for file 3.
- // The second STEPLIB card identifies the load library containing ADARUN and other Adabas execution modules. The field definitions for Adabas file 5 are read from Adabas because no FDTDDNAME has been provided.
- //DDCARD is required. It identifies the ADARUN control cards appropriate for running a batch Adabas program at your installation. The utility uses standard Adabas calls to read Adabas file definitions.
- (...) If Adabas is run in single user mode, other DD cards appropriate to single user mode execution of Adabas are required. Your installation standards specify the appropriate cards.
- //SYSPRINT is required. The SYSPRINT DD statement identifies the output data set to which Unicenter CA-PLEU sends messages and output information.
- //FDTSEXT1 is required by the GLOBALS statement. It identifies the data set that contains the field definitions for Adabas file 3.
- //FDTSEXT3 is required by the GLOBALS statement. It identifies the data set that contains the field definitions for Adabas file 13.
- //SIBA is required. It indicates the Protection Log tape(s) to be processed by the utility.

- //FILEEXT1 specifies the first extract output data set. This data set is on disk, defaults to the maximum blocksize for the device, and holds output from the first extract request. When using devices with track sizes larger than 32760, the default blocksize is 32760. This may result in inefficient use of the disk space, so a smaller blocksize may be appropriate.
- //FILEEXT2 specifies the second extract output data set. This data set is on tape, defaults to the maximum blocksize for the device, usually 32760, and holds output from the second extract request.
- //FILEEXT3 specifies the third extract output data set. This data set is on tape, will use the blocksize specified in the DCB parameter, and holds output from the third extract request.
- //FILEEXTn. One DD statement is required corresponding to each unique DDNAME parameter in the EXTRACT control cards.
- //SYSIN is required. The SYSIN DD statement identifies the source of the input control statements. SYSIN DD * identifies the statements that follow as input. The SYSIN data set may be any sequential file containing control statements in fixed length records.

Note: Although this example shows the first extract directed to disk and the others to tape, this is not a requirement. Any extract can be directed to either disk or tape.

z/OS and OS/390, MSP, and VOS3 Printed Output Data Set

The normal printed output data set is SYSPRINT. The default parameters for this data set are:

- Record format FBA
- Logical record length 133
- Blocksize 1330

The first byte of the record is the ANSI carriage control character. The default record format, blocksize, and logical record length may be altered by the desired values in the DCB parameter of the SYSPRINT DD statement. The minimum logical record size is 133.

Execution in VSE/ESA

Sample JCL

Unicenter CA-PLEU is executed as a standard job. The following is a sample job control deck for VSE/ESA followed by an explanation of each card image.

```
// JOB JOBNAME YOUR INSTALLATION'S ACCOUNTING INFO
// LIBDEF PHASE,SEARCH=(yourlib.PLEUVvvv,ADALIB.ADAxxx),TEMP
// ASSGN SYS000,DISK,VOL=vvvvvv,SHR
// DLBL CARD,'ADABAS.ADARUN',0,SD
// EXTENT SYS000
// ASSGN SYS010,TAPE
// TLBL SYS010,'ADABAS.DATABASE.SIBA',VOL001
// ASSGN SYS021,DISK
// DLBL SYS021,'EXTRACT1.OUTPUT',,SD
// EXTENT SYS021,...
// ASSGN SYS011,TAPE
// TLBL SYS011,'EXTRACT2.OUTPUT'
// ASSGN SYS012,TAPE
// TLBL SYS012,'EXTRACT3.OUTPUT'
    .
    .
    .
// ASSGN SYS0NN,...
// TLBL SYS0NN,'EXTRACTN.OUTPUT',
// EXEC DBGADEXT,SIZE=(AUTO,100K)
GLOBALS ADAVER=71;
SELECT  STARTDATE=19990420  ENDDATE=19990420
        STARTTIME=0800 ENDTIME=1700;
EXTRACT FILE=3 DDNAME=SYS021 FORMAT='AA,BB,...'
        EXTERMID=TRM1,TRM2,...;
EXTRACT FILE=5 DDNAME=SYS011 FORMAT='AA,BB,...'
        USERID=USER1,USER2,...;
EXTRACT FILE=13 DDNAME=SYS012 FORMAT='AA,BB,...'
        BLOCKSIZE=4800 ;
    .
    .
    .
EXTRACT FILE=NN DDNAME=SYS0NN
        FORMAT='AA,BB,...'
        STARTTIME=090000 ENDTIME=120000;
/*
/&
```

VSE/ESA Job Control Language Explanation

Each of the following explanations corresponds to a VSE/ESA control card from above.

- // LIBDEF is optional. The LIBDEF statement identifies the library(s) that contains the DBGADEXT load module and the library containing ADARUN and other modules necessary for Adabas execution.
- // ASSGN SYS000 and associated DLBL and EXTENT cards are required. They identify the ADARUN control cards appropriate for running a batch Adabas program at your installation. If ADARUN is set up for appropriate defaults, SYS000 may be set to IGN. Unicenter CA-PLEU uses standard Adabas calls to read Adabas field definitions.
- // ASSGN SYS010 and the appropriate TLBL or DLBL and EXTENT cards are required. It indicates the Protection Log tape(s) to be processed by the utility.
- // ASSGN SYS021, DLBL and EXTENT cards specify the first extract output data set. This data set is on disk, defaults to an appropriate blocksize for the device, and holds output from the first extract request.
- // ASSGN SYS011 and the appropriate TLBL specify the second extract output data set. This data set is on tape, defaults to an appropriate blocksize for the device, and holds output from the second extract request.
- // ASSGN SYS012 and the appropriate TLBL specify the third extract output data set. This data set is on tape, uses the blocksize specified in the BLOCKSIZE parameter, and holds output from the third extract request.
- An ASSGN statement and the appropriate TLBL or DLBL and EXTENT cards are required corresponding to each unique DDNAME parameter in the extract control cards.
- `/*` identifies the end of the extract control cards in the SYSIPT input stream.

VSE/ESA Logical Unit Assignments

In most cases, the filenames for all data sets are identical to the programmer logical unit names to simplify coding of JCL. The exception to this is the input Protection Log filename, which is different depending on the source, tape or disk, and the read mode, forward or backward. The following list explains the logical unit assignments.

- SYS010 — Input Protection Log

  This logical unit refers to the input Protection Log file when reading the file from tape.

- SYS010B — Input Protection Log

  This logical unit refers to the input Protection Log file when reading the file backwards from tape, specified by the READ-BACKWARDS keyword on the GLOBALS statement.

- SYS010D — Input Protection Log

  This logical unit refers to the input Protection Log file when reading the file from disk. The device type is specified by the SIBA-FROM-DISK keyword on the GLOBALS statement.

- SYS011 through SYS020 — Output Tape Data Sets

  These logical units can be used to write extracts to tape.

- SYS021 through SYS040 — Output Disk Data Sets

  These logical units can be used to write extracts to disk data sets. The user should assure that sufficient space is allocated to handle the volume of records extracted.

- SYS041 through SYS045 — Field Definition Tape Data Sets

  These logical units can be used to identify tape data sets containing the field definitions requested on the GLOBALS statement by the FDTDDNAME keyword.

- SYS046 through SYS050 — Field Definition Disk Data Sets

  These logical units can be used to identify disk data sets containing the field definitions requested on the GLOBALS statement by the FDTDDNAME keyword.

VSE/ESA Printed Output

All printed output is directed to SYSLST.

Chapter 5

Command Reference Information

This chapter is a reference guide for coding and executing control statements in Unicenter CA-PLEU Protection Log Extract Utility for Adabas (Unicenter CA-PLEU).

Unicenter CA-PLEU Command Statements

The command statements are supplied in the form of card images. There are three types of statements:

- GLOBALS statements

  There may be one GLOBALS statement per execution of Unicenter CA-PLEU. It specifies characteristics of the global operating environment for this execution. The GLOBALS statement must be the first statement in the control statement data set. Only comments can precede it.

- SELECT statements

  There may be one SELECT statement per execution of Unicenter CA-PLEU, but it is optional. It specifies global selection criteria that are applied to all records read from the Protection Log. Records that meet these criteria are examined further and perhaps formatted for an output data set. Records that do not meet these criteria are ignored. The SELECT statement, if present, must precede all EXTRACT statements and follow the GLOBALS statement.

- EXTRACT statements

  The mandatory EXTRACT statement has several functions:

  - It defines the format of a single Adabas file record for a single output data set
  - It specifies the selection criteria to be applied to the Protection Log records before allowing them to be written into the output data set
  - It specifies the name of the user exit routine to be called before writing records to the output data set

  A single Protection Log record is tested against all of the EXTRACT statements for a single Adabas file, regardless of the results of any single test. That is, a single Protection Log record may meet the selection criteria on some EXTRACT statements and fail others in a single execution of Unicenter CA-PLEU.

Command Statement Input Stream

The command input stream to Unicenter CA-PLEU consists of a string of comments and command statements. The input stream is sequenced as follows:

1. Any number of comment statements placed in any position
2. A GLOBALS command statement, which describes the global operating environment
3. An optional SELECT command statement, which sets global selection criteria for the execution
4. One or more EXTRACT command statements, which define the Protection Log extracts to be output

You can stop input card-images from printing on the output message data set by placing a percent sign, `%`, in the first column of the card image. This is useful primarily to avoid displaying passwords in the PASSWORD parameter of the GLOBALS statement.

Command Statement Comments

Comments may be inserted anywhere in the input stream. They may contain arbitrary text.

Text is identified as comments in one of two ways:

- An asterisk, "*", in the first column of the input record identifies the record as a comment.
- The characters "/*" or "//" identify the remainder of the record as comment and cause the command interpreter to skip to the next record. "/*" and "//" may not be placed in column 1 of the control data set. If coded in column 1, they are interpreted as JCL cards and thereby end the input stream.

Comment Examples

The following are examples of comments:

```
* This is a comment
SELECT STARTDATE=20010101   /* Happy new year
ENDDATE=20010131;           // End of the month
```

Command Statement

The command statements for Unicenter CA-PLEU begin with a statement identifier keyword and are terminated by a semi-colon, "`;`", the next statement identifier keyword, or the end of the command input file. Detailed information on the format of these statements is given on the following pages.

Statement Elements

Statement elements are the components of statements and are separated by spaces. Place commas between statement elements only where specified.

Parameter-Value Lists

Certain parameters take as values a single word, a list of single words or a group of parameter lists. Each simple entry is equivalent to a character string, but does not require surrounding apostrophes. Quoted and hexadecimal character strings are acceptable as list entries.

Parameter list entries are separated by commas, `","`, and the list ends with the entry not followed by a comma. Parameter lists may optionally be surrounded by parentheses. Their use can improve clarity for the user when coding groups of parameter lists. The following example specifies a list of three terminal-IDs:

- S193
- A terminal with an unprintable hexadecimal identifier
- S195

```
TERMID=S193,H'FF000044',S195
```

Parameter value lists may be continued across command record (card image) boundaries by ending the list on the first record with a comma and continuing the list in any position on the next card. The individual values must not be coded across record boundaries.

```
USERID=usera1,usera2,usera3,
userb1,userb6,userb7
```

The following example specifies a list of three DDNAMEs to be used for reading field definitions for specific Adabas files:

```
FDTDDNAME=(DDNAME,(3,DDNAME2),
(5,DDNAME3))
```

Numeric Constants

Numeric constants may be fixed integer or decimal.

- Fixed Numeric Constants: Fixed integer constants do not have decimal points. They may be coded as either decimal or hexadecimal numbers. The largest integer that may be entered is 2,147,483,647; the smallest integer that may be entered is -2,147,483,648.
- Decimal Fixed Integer Constants: A fixed integer constant may be entered as a number consisting of decimal digits 0 through 9 preceded by an optional sign "+" or "-".

  ```
  456
  -21456
  +284
  ```

Character String Constants

Character string constants are entered between apostrophes, for example, '`string_constant`'. The character string is made up of any characters in the collating sequence. The character string may be up to 254 characters long.

Alphanumeric Character Strings

An alphanumeric character string is made up of any character in the collating sequence.

```
'Computer Associates International, Inc.'
```

Hexadecimal Character Strings

A hexadecimal character string is entered as a string of an even number of hexadecimal digits (0 thru F) preceded by the letter H and enclosed in apostrophes, " ' ".

```
H'C1C2C3' H'FFFFFFFF'  H'0000'
```

Continuing Character Strings

Character string constants cannot be continued across line boundaries; however, they can be continued by use of a concatenation operator between them. The concatenation operator for character strings is a dash, "-".

For example, the following character strings are equivalent.

```
'ABC' - H'C4' - 'EF'
* and
'ABCDEF'

'THIS IS A CHARACTER STRING' -
'ACROSS AN INPUT LINE'
* and
'THIS IS A CHARACTER STRING ACROSS AN INPUT LINE'
```

Date Constants

A date should be entered as *yyyymmdd* where *yyyy* is the four digits of the year, *mm* is the month, and *dd* the day.

```
'20020322' '20010101'
```

Time Constants

A time should be entered as *hhmmss*, where *hh* is the hour of the day as indicated by a 24-hour clock, *mm* is the minutes, and *ss* is the seconds of the time.

```
'123000' '123059' '235959' '000000'
```

GLOBALS: Definition of Global Environment

Syntax Format

The syntax of the GLOBALS statement is as follows:

```
GLOBALS
 [ ADAVER = {71 | 72 | 74 } ]
 [ CPU-ID = cpuid ]
 [ DBNAME = dbname ]
 [ FDTDDNAME =
   ( { ddname
     | (file_nmbr_1,ddname_1) [,…,(file_nmbr_n,ddname_n)]
     | ddname,(file_nmbr_1,ddname_1) [,…,(file_nmbr_n,ddname_n)] }
   ) ]
 [ LOCALTIME = ( { E | W } number_timezones ) ]
 [ PAGESIZE = {55 | number} ]
 [ PASSWORD =
   ( { password
     | (file_nmbr_1,password_1) [,…,(file_nmbr_n,password_n)]
     | password,(file_nmbr_1,password_1)
         [,...,(file_nmbr_n,password_n)] }
   ) ]
 [ READ-BACKWARD = {YES | NO} ]
 [ SIBA-FROM-DISK = {YES | NO} ]
;
```

The following two parameters are applicable only to Adabas version 4. If they appear in the GLOBALS statement, they are simply ignored; no error is generated.

```
DBID = dbid
LONG-REC-HEAD = {YES | NO}
```

Note: The SIBA-FROM-DISK parameter applies only to VSE/ESA.

Syntax Example

Each execution of Unicenter CA-PLEU must begin with a single GLOBALS statement. This has the effect of defining the environment for this execution of the utility.

```
GLOBALS ADAVER=74
        FDTDDNAME=(SAVEFDTS,(3,SAVFDT03),
        (10,SAVFDT10))
        LOCALTIME=W9
        PAGESIZE=50
        PASSWORD=(READOKAY);
```

ADAVER Parameter

```
ADAVER = { 71 | 72 | 74 }
```

The optional ADAVER parameter specifies the Adabas version that created the Protection Logs. The value specified can be 71, 72, or 74. The default value of 71 is applied if the ADAVER parameter is not specified.

CPU-ID Parameter

```
CPU-ID = cpuid
```

The optional CPU-ID parameter specifies an eight (8) byte character string that is inserted into the OUT5CPID field in the output record header. It can be used to identify the source of the data.

DBID Parameter

```
DBID = dbid
```

This GLOBALS parameter is applicable only to Adabas version 4 and is now obsolete since Unicenter CA-PLEU no longer supports Adabas version 4.

This parameter has no effect because the DBID is taken directly from the Protection Log record. If present, the parameter is ignored.

DBNAME Parameter

```
DBNAME = dbname
```

The optional DBNAME parameter specifies a string of eight characters or less that is inserted into the OUT5DBNM field in the output record header. It can be used to differentiate between various databases.

FDTDDNAME Parameter

```
FDTDDNAME =
( { ddname
  | (file_nmbr_1,ddname_1) [,…,(file_nmbr_n,ddname_n)]
  | ddname,(file_nmbr_1,ddname_1) [,…,(file_nmbr_n,ddname_n)] }
)
```

The optional FDTDDNAME parameter directs Unicenter CA-PLEU to read Adabas field definitions from specific sequential data sets created by the SAVEFDTS program. If not specified, Unicenter CA-PLEU reads the field definitions for an Adabas file directly from the database using an LF command. The FDTDDNAME keyword can be used in three ways:

- To read all field definitions from a single data set
- To read the field definitions for individual Adabas files from separate data sets
- To read all field definitions from a single data set as well as field definitions for specific Adabas files from other data sets

FDTDDNAME Examples

The following examples illustrate how each of these options might be used:

- The Protection Log to be read was created prior to several file modifications that make the online field definitions incompatible with the Protection Log. Prior to making the field definition changes, the field definitions were written to a sequential data set by the SAVEFDTS utility. This data set is referred to by DD or file name FDTFILE. This execution of the utility uses the definitions found in this data set for all of the Adabas files being extracted.

  ```
  GLOBALS FDTDDNAME=FDTFILE ;
  ```

- The field definitions for Adabas file 10 are read from the data set identified by DD or file name FDT10. The field definitions for Adabas files 15 and 16 are read from the data set identified by DD or file name FDT15. All other field definitions are read from the online Adabas files using the LF command.

  ```
  GLOBALS FDTDDNAME=((10,FDT10),(15,FDT15),(16,FDT15)) ;
  ```

- The field definitions for Adabas files 7, 8 and 9 are read from the data set identified by DD or file name FDT789. The field definitions for Adabas files 15 and 20 are read from the data set identified by DD or file name FDT1520. All other field definitions are read form the data set identified by the DD or file name FDTREST.

  ```
  GLOBALS FDTDDNAME=(FDTREST,
                     (7,FDT789),(8,FDT789),(9,FDT789),
                      (15,FDT1520),(20,FDT1520)) ;
  ```

LOCALTIME Parameter

LOCALTIME = {*E*|*W*} *number_timezones*

The optional LOCALTIME parameter is used to convert the GMT timestamp that Adabas puts in each PLOG record to avoid ambiguity to the user’s local time.

Valid LOCALTIME values contain an E or W in the first position followed by the number of time zones, 1 through 12, in a given direction. “E” indicates a time zone East of Greenwich and “W” for time zones West of Greenwich.

For example, the following statement subtracts 5 hours from the PLOG timestamp:

```
LOCALTIME = W5
```

And this example adds 2 hours to the reported time:

```
LOCALTIME = E2
```

LONG-REC-HEAD Parameter

```
LONG-REC-HEAD = {YES|NO}
```

This GLOBALS parameter is applicable only to Adabas version 4 and is now obsolete since Unicenter CA-PLEU no longer supports Adabas version 4.

The LONG-REC-HEAD parameter is ignored if specified.

PAGESIZE Parameter

```
PAGESIZE = {55 | number}
```

The optional PAGESIZE parameter specifies the number of lines per page on the print data set. The value should be specified as a positive decimal number. For example, the following statement results in a new page carriage control being inserted after 45 lines have been used on a page.

```
GLOBAL PAGESIZE=45
```

The default value is 55 lines and is applied when PAGESIZE does not appear in a GLOBALS statement.

PASSWORD Parameter

```
PASSWORD = ( { password
             | (file_nmbr_1,password_1)
               [,…,(file_nmbr_n,password_n)]
             | password,(file_nmbr_1,password_1)
                       [,…,(file_nmbr_n,password_n)] }
           )
```

Unicenter CA-PLEU reads the field definitions for an Adabas file using an LF command unless otherwise directed by the FDTDDNAME keyword on the GLOBALS statement. Older releases of Adabas require a read password for the LF command for password-protected files in order to read file definitions. The keyword is retained for compatibility.

The necessary passwords are supplied by the PASSWORD parameter.

Note: The printing of control cards containing the PASSWORD parameter values is suppressed by placing a percentage sign, %, in the first column of the input card.

The PASSWORD parameter has a similar syntax to the FDTDDNAME parameter. A default password may be specified, followed by a list of passwords for specific files. Examples of use of the parameter follow:

- Use the password HYANDMYT for all LF commands to the database. This must be a global read-access password for all security-protected files being processed by the utility.

  ```
  % GLOBALS PASSWORD=HYANDMYT ;
  ```

- Files 10, 15 and 16 are password-protected; no other files being processed are password-protected for read access.

  ```
  GLOBALS PAGESIZE=60
  % PASSWORD=( (10,PASSWD10),(15,SECRET),
  % (16,CONFDTL) ) ;
  ```

- The password for Adabas files 7, 8 and 9 is PERSONL. The password for Adabas files 15 and 20 is SALARY. The password for general read-access to other files is GENERAL.

  ```
  %GLOBALS PASSWORD=(GENERAL,
  % (7,PERSONL),(8,PERSONL),(9,PERSONL),
  % (15,SALARY),(20,SALARY) ) ;
  ```

READ-BACKWARD Parameter

```
READ-BACKWARD = {YES | NO}
```

The READ-BACKWARD parameter instructs Unicenter CA-PLEU to process a Protection Log tape in reverse sequence. This allows extraction of data from most recent to least recent images. This may be used, for example, to extract only the last before image of a record prior to its deletion or only the most recent after image of a record for update of a shadow database.

The tape is opened at end-of-file and processed backwards if the following keyword is included in the GLOBALS statement:

```
READ-BACKWARD=YES
```

SIBA-FROM-DISK Parameter

```
SIBA-FROM-DISK = {YES | NO}
```

The optional SIBA-FROM-DISK parameter is used only in VSE/ESA. The utility reads the Protection Log from the tape unless the following keyword entry requests reading from disks.

```
SIBA-FROM-DISK = YES
```

All other operating systems can read Protection Log records from either disk or tape without using this keyword.

EXTRACT: Build an Extract Output File

Syntax Format

The manadatory EXTRACT statement has the following syntax.

```
EXTRACT
 DDNAME = ddname
 FILE = adabas_file_number
 [ BLOCKSIZE = blocksize ]
 [ [EX]GENERAL-ID        = (id_1[,…,id_n])         ]
 [ [EX]GENERAL-ID-CPU-ID = (cpuid_1[,…,cpuid_n])   ]
 [ [EX]GENERAL-ID-OS-ID  = (osid_1[,…,osid_n])     ]
 [ [EX]GENERAL-ID-USERID = (userid_1[,…,userid_n]) ]
 [ [EX]GENERAL-ID-VM-ID  = (vmid_1[,…,vmid_n])     ]
 [ EXITNAME = module_name ]
 [ [EX]TERMID = (term_1[,…,term_n]) ]
 [ [EX]USERID = (user_1[,…,user_n]) ]
 [ FORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD }
   | AFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
   | BFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
   | DFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
   | NFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
   | SFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
   | UFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
 ]
 [ ISN = H'hex_value' ]
 [ RECSIZE = maximum_record_size ]
 [ STARTDATE = yyyymmdd ] [ ENDDATE = yyyymmdd ]
 [ STARTTIME = hhmmss ] [ ENDTIME = hhmmss ]
 [ TSN = H'hex_value' ]
 [ USERDATA = {YES | 'character_string'} ]
;
```

Note: The BLOCKSIZE parameter is specified only in VSE/ESA.
The USERDATA parameter is specified only when extracting C1, C3, C5, or ET type records.

Syntax Example

The EXTRACT statement describes an extract file to be built during Unicenter CA-PLEU execution. At least one EXTRACT statement must be specified, although any number can be used.

```
EXTRACT DDNAME=FILEEXT1
        FILE=3
        EXITNAME=EXIT03
        FORMAT='AA,12,U,CC1-6,6,VC1-20.'
        RECSIZE=156
        STARTDATE=20010701 ENDDATE=20010701
        EXUSERID='ACCOUNT1','PERSONL3';
```

The EXTRACT statement specifies the following:

- The Adabas file for which the extract is to be made
- The format of the extracted output records, specified by an Adabas format buffer or the keywords ADDFORM, ADDFORM-LONG, or UNLOAD
- The maximum size of the output records
- The output file to which the extracted data is to be written
- Whether before images, after images or both are to be written

- Additional selection criteria beyond those specified by the SELECT statement
- The name of the exit routine to be given control prior to writing the formatted record to the output file

BLOCKSIZE Parameter

BLOCKSIZE = *blocksize*

The optional BLOCKSIZE parameter is be used to override the default blocksize for disk or tape units. It is specified only for VSE/ESA.

In z/OS and OS/390, the same operation is accomplished by coding the DCB=BLKSIZE parameter in the JCL.

If this parameter is specified in more than one EXTRACT for a given DD or file name, the largest blocksize is used.

DDNAME Parameter

DDNAME = *ddname*

The mandatory DDNAME parameter specifies the output file to which the extract is to be written.

When two or more EXTRACT statements specify the same ddname, the formatted output records created by each statement is merged on the output data set in the order in which the corresponding records are read.

In z/OS and OS/390, DDNAME specifies the name of one of the DD statements specified in JCL.

In VSE/ESA, DDNAME specifies a SYS*nnn* logical-unit name to which the extract is to be written.

EXITNAME Parameter

EXITNAME = *module_name*

Each EXTRACT statement can specify its own exit routine. Thus, each exit routine can be written assuming a single Adabas file number, a single set of record selection criteria and a single output data set. The load module name specified as the value of this parameter is loaded at the start of Unicenter CA-PLEU execution. The routine is given control prior to writing a record for this extract.

Several EXTRACT statements can reference the same exit routine. The routine is called whenever a record is written for any of the several EXTRACT statements. Because only a single copy of the exit is loaded, the routine must have the ability to differentiate between the record types it will process.

Note: When a formatting parameter is used to apply the ADDFORM or ADDFORM-LONG to records, a user exit must be specified. These two formats generate records of variable format which must be processed by the user exit that has access to the format buffer.

Techniques for coding the user exits are discussed in the "User Exits" chapter of this document. The formatting parameters are discussed below.

FILE Parameter

FILE = *adabas_file_number*

This mandatory parameter specifies the Adabas file whose images are to be extracted. It is the standard file number between 1 and 65535.

Format Parameters

```
FORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD }
| AFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
| BFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
| DFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
| NFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
| SFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
| UFORMAT = {adabas_format | ADDFORM | ADDFORM-LONG | UNLOAD}
```

These parameters specify the format in which records are written to the file specified by DDNAME.

- FORMAT specifies the format for this extract in which the before and after images for all records are written
- AFORMAT specifies the format for this extract in which the after images for all records are written
- BFORMAT specifies the format for this extract in which the before images for all records are written
- DFORMAT specifies the format for this extract in which the before images of deleted records are written
- NFORMAT specifies the format for this extract in which the after images of newly added records are written
- SFORMAT specifies the format for this extract in which the before images of deleted records and the after images of newly added records are written
- UFORMAT specifies the format for this extract in which the before and after images of updated records are written

All seven parameters offer the same formatting selections.

Adabas Format

```
field_name [,length][,format] . . .
```

where *field_name* is:

```
field_name { [i[-j][(m[-n])]]
           | N | 1-N | N(N) | N(1-N)
           }
```

adabas_format specifies a fixed format; for example, as input to a program written in COBOL or Natural. The value should be coded using the rules for Adabas direct call format buffers.

Note: Users are cautioned that the format of each extract file is defined by the format specifications entered in the EXTRACT statement. Users not familiar with the Adabas syntax should consult the *Adabas Command Reference Manual* before attempting to code format specifications in Unicenter CA-PLEU.

Adabas-format Specification

The rules for the output format specifications are those specified in the *Adabas Command Reference Manual* with these exceptions:

- Edit formats, E1 through E15, are not supported
- The repetitive form of specifying multiple value fields (MF,MF,MF instead of MF1-3) is not supported
- Literals and nX specifications are not supported
- Range specifications, such as F1-F2, are not supported

Adabas-format Syntax

If you are addressing an Elementary field, you need only specify the fieldname and optionally a length and format.

```
field_name[,length][,format]
```

If you are addressing an MU or a PE field, you would specify the fieldname and, optionally, a number of occurrences i through j, and an optional length and format.

```
field_name[i[-j]][,length][,format]
```

If you are addressing an MU field within a PE field, you would specify the name of the MU field and, a number of occurrences i through j of the PE field in which the MU field resides, optionally the number of occurrences of the MU field, m though n entered within parentheses, and finally, an optional length and format.

```
field_name[i[-j][(m[-n])]][,length][,format]
```

Please note that you cannot specify the occurrences of an MU field within a PE field without specifying at least one occurrence of the PE field.

Unicenter CA-PLEU also offers **1-N** and **N** that allow you to extract all or only the last occurrences of MU and PE fields. These keywords are specified instead of the numbers representing occurrences.

```
Field_nameN[,length][,format]
```

extracts the last occurrence of an MU field.

```
Field_name1-N[,length][,format]
```

extracts all occurrences of an MU or PE field.

```
Field_nameN(N)[,length][,format]
```

extracts the last occurrence of an MU field in the last occurrence of a PE field.

```
Field_nameN(1-N)[,length][,format]
```

extracts all occurrences of an MU field in the last occurrence of a PE field.

Format and Length

Adabas formats may be any of the following: A, B, F, G, P, or U. Length must conform to Adabas rules. If neither length nor format is specified, the original length and format remain.

Examples

```
FORMAT='AA,BB,CC.'
```

extracts the fields AA, BB and CC, leaving their original length and format.

```
BFORMAT='AA,5,A,BB,3,MFC,3,U,MF1-6,6,U.'
```

extracts the field AA, applying a length of five and the A format; the field BB, applying a length of three; the field MFC, applying a length of three and the U format; and occurrences 1 through 6 of the MF field applying a length of 6 and the U format.

```
AFORMAT='CB1-2(1-4).'
```

extracts first through fourth occurrences of the MU field CB and the first two occurrences of the PE in which CB resides.

```
FORMAT='AQC,3,U.'
```

extracts the number of occurrences in the count field of AC, sets a length of three and the U format.

```
BFORMAT='AQ1-N,3,U.'
```

extracts all occurrences of the PE or MU field AQ, sets a length of three and the U format.

```
AFORMAT='ATN(1-N),5,U.'
```

extracts all occurrences of MU field AT in the last occurrence of the PE and sets a length of 5 and the U format.

Note: You cannot instruct Unicenter CA-PLEU to extract all occurrences of an MU field and all occurrences of a PE. That is, the function ***`field_name1-N(1-N)`*** is not supported in this release.

ADDFORM and ADDFORM-LONG Formats

The ADDFORM and ADDFORM-LONG parameter values for the Format parameters of the EXTRACT request are designed to be used together with a user exit in building an audit database.

An audit database consists of files with the same file definitions as the production database and additional fields corresponding to the Protection Log information; for example, before/after, terminal-ID, date/time, etc. With this information, Natural or other query systems can generate reports of what fields have been modified, who has modified them and so forth. The audit database can be built directly from the Unicenter CA-PLEU using a special user exit which adds audit records directly to the audit database.

Output of ADDFORM and ADDFORM-LONG

When the ADDFORM or ADDFORM-LONG format is specified for an EXTRACT, the output consists of two components:

- The decompressed records
- Format buffers specifying the format of the decompressed records

The decompressed record contents are variable format and consist of only non-null fields. This results in considerable saving in output record length for Adabas files with multiple logical record types achieved by null-suppression, or files with periodic groups with large possible maximum occurrences but a small number of average occurrences. The Adabas dictionary file is a prime example of this.

ADDFORM Format

When FORMAT=ADDFORM is specified, all non-null fields, including blank and zero fields that are not null-suppressed, are output with default length and format as specified in the FDT. A message is issued if this results in field truncation.

ADDFORM-LONG Format

When this is specified, all non-null fields, including blank and zero fields that are not null-suppressed, are output in their actual length rather than with the length specified within the FDT. The format buffer that accompanies each decompressed record contains the actual length of each field. Since Adabas deletes leading zeroes or trailing blanks from fields that are not fixed length, the actual field length is often less than the length specified in the FDT. In addition, some applications store fields that are longer than the length specified in the FDT. ADDFORM-LONG decompresses those fields without risk of truncation.

The data portion of the decompressed record and the format buffer that are built by Unicenter CA-PLEU are presented in a form so that they can be used directly as the record buffer and the format buffer in an Adabas direct call. Some of the other fields of the Adabas Control Block, for example, the ISN and Adabas file number, can be extracted from the header portion of the decompressed record.

Since the length of each field is dependent on the data contained within it, the length of each field can be expected to vary from one record to another. The format buffer must be used to extract data from the record. One must “step through” the decompressed data, using the lengths found in the format buffer that accompanies the decompressed record.

Periodic Groups

For periodic groups, when ADDFORM is specified, null values are included for all fields in a group containing any non-null value, to allow specification of the group name in the format buffer. When ADDFORM-LONG is specified, the fields are put out with individual format buffer entries, and null values are not included.

If a periodic group contains a multiple-value field, all fields in the periodic group are individually defined, and no null values are included in the periodic group.

Format Buffers

Unicenter CA-PLEU builds a format buffer for each record put out in ADDFORM or ADDFORM-LONG format. This format is passed to the user exit, and may be processed there. The form of the format buffer varies depending on the specification of ADDFORM or ADDFORM-LONG.

ADDFORM Format Buffers

The ADDFORM format buffer assumes that FDT default definitions are satisfactory and creates as short a format buffer as possible within the constraints of varying output record formats. The format buffer built has the following characteristics:

- Simple fields are defined in the format buffer as "AA,"; the default field format and length are assumed to hold.
- Periodic groups not containing multiple-value fields are defined as "PGn-m" where n and m are the first and last non-null occurrences in the decompressed record.
- Multiple-value fields are defined in the format buffer as "MU1-m,", where m is the maximum occurrence in the record that has been decompressed.
- Periodic groups containing multiple-value fields are decompressed "PA1,PB1,PM1(1-3),PA2,PB2,PM2(1),..." with an individual definition for each field in the periodic group and a range for multiple-value fields.

ADDFORM-LONG Format Buffer

The ADDFORM-LONG format buffer assumes nothing about the formats of fields, and individually defines every field in the output record. The format buffer built has the following characteristics:

- Simple fields are defined in the format buffer as "AA,l,f"; l is the field length and f is the field format.
- Periodic groups fields are individually defined with explicit length and format; that is, "PFn,l,f"; n is field occurrence number, and l and f are length and format.
- Multiple-value fields are defined in the format buffer as "MU1-m,l,f", where m is the maximum occurrence, l and f are the length and format.
- Periodic groups containing multiple-value fields are decompressed as in the example above; for example, "PA1,5,A,PB1(1-2),2,U,PA2,5,A,...", with an individual definition for each field in the periodic group and a range for each multiple-value field.

Using ADDFORM Formats

The ADDFORM formats can be used to add records to an audit database by writing a special user exit as outlined below. A single user exit program can be used for all files in the database. Users should be aware of the possible performance implications of such a large batch update process.

Logic of Record Add Exit

The following is an example of the steps needed to build the audit record database:

1. On the initial call, open the audit database. EXU usage for all files being processed is recommended.
2. Each time the exit is called, build an Adabas Control Block and buffers as follows:
   - Command-code: "N1" is Add.
   - Command-ID: spaces, formats vary.
   - File-number: file number in parameter list (put the database-ID of the audit database in the first byte if running in MPM mode); translate the file number if the audit numbers are different from the production file numbers.
   - For Adabas version 6 and higher, place X'30' in the first byte of the Adabas Control Block, put the 2-byte file number in the file number field, and the 2-byte DBID in the response code field.
   - Format-buffer-length: length supplied by Unicenter CA-PLEU + header length.
   - Record-buffer-length: length supplied by Unicenter CA-PLEU.
   - Format-buffer: append format buffer supplied by Unicenter CA-PLEU to a standard format buffer definition of the record header.
3. Call Adabas to perform the Add. Count records, etc.
4. On the final call, close the database and print record counts, etc.

Note: If you run single user mode, be sure to use SAVEFDTS file definitions if running against a different database than the one that created the Protection Log. Otherwise, Unicenter CA-PLEU gets its file definitions from the active database.

UNLOAD Format

Output is desired in a format that is similar to that produced by the ADASEL utility. For records produced, the Unicenter CA-PLEU record header is included.

A Unicenter CA-PLEU header contains all header items provided in the ADASEL LOGINFO header, plus more. However, items in the Unicenter CA-PLEU headers are not in the same sequence as in the ADASEL headers.

RECSIZE Parameter

RECSIZE = ***maximum_record_size***

The RECSIZE parameter is used to specify the maximum record size in bytes that is produced for this extract. This parameter is used when one or more of the following situations exists.

- A user exit that appends data to the output records is specified in this EXTRACT statement.
- A user exit that inserts records that are longer than the extract records being produced by Unicenter CA-PLEU is specified in this EXTRACT statement.
- The ADDFORM, ADDFORM-LONG, or UNLOAD format has been applied in the EXTRACT statement and default buffer size estimate calculated by the utility is inadequate.

If Adabas field descriptions are being used for the format specification and no records are being inserted or appended by a user exit, the RECSIZE parameter is not needed.

Unicenter CA-PLEU estimates the record size required for the format defined by the value of the Format parameter. Unless overridden by RECSIZE, this estimate is used when allocating buffer space for building the output records. Several situations could make this estimate too small and the RECSIZE parameter allows the user to inform the utility that allowance must be made for larger records.

The estimate made for an ADDFORM , ADDFORM-LONG, or UNLOAD format can be insufficient if some periodic groups or multiple-value fields have a large number of occurrences.

In such a case, the compressed record stored within Adabas could be very short while the decompressed record, with all of the null values expanded to their full lengths, could be very large. If you expect such a case, estimate the size of the largest possible decompressed record and specify that value with the RECSIZE parameter.

It is also possible that the length of a field within the compressed record is greater than the default length given in the FDT. For example, the FDT may give a default length of eight for a character string but the actual string is 100 characters long. As far as Adabas is concerned, this is not an error, it can process the string correctly.

However, when the ADDFORM, ADDFORM-LONG or UNLOAD formats are used, the FDT specifications are used and Unicenter CA-PLEUs estimate of the record size may be too small. Two approaches can be used to solve this problem.

One could supply a value of RECSIZE, specifying a value large enough to allow for the actual lengths of all fields. Alternatively, one could revise the FDT so that the default field lengths are set to the maximum field lengths.

The RECSIZE parameter may be required when the format specification is made up of Adabas field descriptions. The record size computed by Unicenter CA-PLEU is large enough for the record described by the format specification. However, this record size may be too small if a user exit routine applied to this extract inserts records or appends information onto records generated. The user should estimate the size of the largest possible record being inserted and the largest possible record after appending information. This value should be used as the value for RECSIZE.

If the exit routine is inserting records but not appending data onto existing records, the user should specify a value of RECSIZE equal to the largest possible inserted record. If the record size required by the format specification is larger than the value of RECSIZE, the larger value is used when allocating buffer space. Therefore, no harm is done if the value of RECSIZE is less than the record size required by the format specification.

Note: The buffers allocated for building each EXTRACT's output records are independent of those allocated for other EXTRACT statements. This is true even if the output records are written to the same output file. The size of the buffers allocated for one EXTRACT, due to the estimated length or to the value of RECSIZE , does not affect sizes for other EXTRACTs. Do not assume a large RECSIZE for one EXTRACT results in sufficient buffer space for all EXTRACTs.

C5, ET, C3, and C1 Records

Records produced by C5, ET, and C3 commands may also be extracted by Unicenter CA-PLEU. Data associated with C5, ET, and C3 commands may otionally be extracted. This data may be written to any output data set, either a separate data set, C5 data for example, or to a data set shared by other EXTRACT requests. ET records may be written to a number of different output files to delimit logical transactions.

C5 data is usually written for application-specific audit trail or recovery purposes.

ET records are important in indicating the logical completion of transactions. ET data may be of interest for audit trail purposes. If logical transaction boundaries are important, the ET data may be written to the same output files as data records, and the data sorted by user-ID, terminal-ID, and transaction sequence number. This gives a picture of the activity of each individual user (job or interactive session).

User data written from C5, ET, and C3 records may be selected based on the first 1 to 127 characters in the data.

These special types of records are selected using standard EXTRACT statements. The record type is specified instead of the file number. The Format parameter is not specified.

```
EXTRACT FILE=C5 DDNAME=C5DATA TERMID=S183
```

USERDATA Parameter

```
USERDATA = { YES | 'character_string'}
```

Data is selected for ET, C5, and C3 records using the USERDATA parameter. The parameter may be used either to indicate that data should be written out for all records by specifying:

```
USERDATA=YES
```

Alternatively, the user can specify that only certain records be written out by entering:

```
USERDATA='character_string'
```

In this case, the character string is compared to the same number of characters in the data portion of the record. The record is written only if the characters are equal.

There is no way to distinguish ET and C3 records. Users should therefore request ET records.

```
EXTRACT FILE=ET DDNAME=PERSDATA USERDATA='PERSONNEL APPL';
```

Selection Criteria Parameters

EXTRACT and SELECT statements offer identical selection criteria parameters.

Selection may be on the basis of:

- An inclusive list, which selects only the records associated with the units specified in the list.
- An exclusive list, which selects only the records associated with units not specified in the list.
- Units can be one or more terminal-IDs, user-IDs, General Unique Identifiers, the GENERAL-ID, or any of four elements that make up the General Unique Identifier. These elements include the CPU-ID, VM-ID, OS-ID, and USERID, each of which is prefaced with the term GENERAL-ID, for example, GENERAL-ID-CPU-ID. One or more of these elements can be specified.
- A period of time, which selects only records created between a starting date and time and an ending date and time.

- ISN, which selects only the records associated with a given Internal Sequence Number.
- TSN, which selects only the records associated with a given Transaction Sequence Number.

If it suits your needs, you can specify all necessary selection criteria in an EXTRACT statement and omit a SELECT statement.

However, when a SELECT statement is used, the selection criteria for the EXTRACT are logically ANDed to those in the SELECT statement. That is, the SELECT criteria are checked before the EXTRACT statement selection criteria. A record is not included in an extract if it fails the SELECT selection criteria.

In other words, it is possible to use the EXTRACT statement to refine or alter the selection criteria in the SELECT statement. For example, a SELECT statement might specify a period of time as a selection criterion. An EXTRACT statement, as well as specifying a ddname, an Adabas file, and a format, might then specify particular users, CPUs, or terminal-IDs as further selection criteria. The result would be the extraction of records within the time period specified by the SELECT statement, but restricted to the records associated with the users, CPUs or terminal-IDs specified by the EXTRACT statement.

Note: The user should be aware that EXTRACT criteria could contradict SELECT criteria. Mutually exclusive selection criteria can produce an empty extraction file. For example, using a SELECT statement to select the records associated with users A, B, and C and then an EXTRACT statement to extract the records associated with all users except users A, B, and C. Specifying such criteria generates no error.

Inclusive and Exclusive Lists

The following parameters specify either an inclusive list or, when prefaced with the letters EX, an exclusive list. An inclusive list selects records associated with the units specified in the list; an exclusive list selects records associated with units not in the list.

The exclusive list is provided to allow the user of Unicenter CA-PLEU to bypass updates made by expected Adabas users and identify those updates made by unexpected users.

Note: Inclusive and exclusive lists, addressing the same unit, are mutually exclusive. For example, the same SELECT statement cannot include both the inclusive GENERAL-ID and the exclusive EXGENERAL-ID. However, nothing prevents you from specifying the same parameter more than once or specifying an inclusive list that addresses one type of unit and an exclusive list that addresses another type.

[EX]GENERAL-ID Parameter

```
GENERAL-ID=(id_1[,…,id_n])
EXGENERAL-ID=(id_1[,…,id_n])
```

The GENERAL-ID and EXGENERAL-ID parameters specify an inclusive or an exclusive list of General Unique Identifiers. The 28-byte General Unique Identifier is the CQEUSID field of the Adabas Command Queue Element (CQE).

[EX]GENERAL-ID-CPU-ID Parameter

```
GENERAL-ID-CPU-ID=(cpuid_1[,…,cpuid_n])
EXGENERAL-ID-CPU-ID=(cpuid_1[,…,cpuid_n])
```

The GENERAL-ID-CPU-ID and EXGENERAL-ID-CPU-ID parameters specify an inclusive or an exclusive list of the eight-byte CPU-ID sub-field of the General Unique Identifier. This sub-field is the CQECPUID field of the Adabas Command Queue Element (CQE).

[EX]GENERAL-ID-OS-ID Parameter

```
GENERAL-ID-OS-ID=(osid_1[,…,osid_n])
EXGENERAL-ID-OS-ID=(osid_1[,…,osid_n])
```

The GENERAL-ID-OS-ID and EXGENERAL-ID-OS-ID parameters specify an inclusive or an exclusive list of four-byte OS identifiers, a sub-field of the General Unique Identifiers. This sub-field is the CQEOSID field of the Adabas Command Queue Element (CQE).

[EX]GENERAL-ID-USERID Parameter

```
GENERAL-ID-USERID=(userid_1[,…,userid_n)]
EXGENERAL-ID-USERID=(userid_1[,…,userid_n)]
```

The GENERAL-ID-USERID and EXGENERAL-ID-USERID parameters specify an inclusive or an exclusive list of eight-byte USERIDs, a sub-field of the General Unique Identifier. This sub-field is the CQEUID field of the Adabas Command Queue Element (CQE).

[EX]GENERAL-ID-VM-ID Parameter

```
GENERAL-ID-VM-ID=(vmid_1[,…,vmid_n])
EXGENERAL-ID-VM-ID=(vmid_1[,…,vmid_n])
```

The GENERAL-ID-VM-ID and EXGENERAL-ID-VM-ID parameters specify an inclusive or an exclusive list of eight-byte virtual machine (VM) identifiers, a sub-field of the General Unique Identifier. This sub-field is the CQEVMID field of the Adabas Command Queue Element (CQE).

[EX]TERMID Parameter

```
TERMID=(termid_1[,…,termid_n])
EXTERMID=(termid_1[,…,termid_n])
```

The TERMID and EXTERMID parameters specify an inclusive or an exclusive list of terminal-IDs. The terminal-IDs must be specified in the format recognized by Adabas. The id is four or fewer characters; values depend on the TP monitor and the network standards.

[EX]USERID Parameter

```
USERID=(userid_1[,…,userid_n])
EXUSERID=(userid_1[,…,userid_n])
```

The USERID and EXUSERID parameters specify an inclusive or an exclusive list of user-IDs. User-IDs are the eight-character user identifiers supplied by the application program in the Adabas Control Block for use in identifying ET data, setting user priority, and generally identifying the user in a terminal-independent fashion. If no user-IDs are supplied by the application, this parameter may not be used.

STARTDATE ENDDATE Parameters

STARTTIME ENDTIME Parameters

```
STARTDATE=yyyymmdd  [ ENDDATE = yyyymmdd ]
STARTTIME=hhmmss  [ ENDTIME = hhmmss ]

Valid time values: 000000 thru 235959
```

These two pairs specify the beginning and the ending date or time. Records whose creation date and time is after the date and time specified by the STARTDATE and STARTTIME parameters and records whose creation date and time is before the date and time specified by the ENDDATE and ENDTIME parameters are selected for extraction.

The user can specify neither, either, or both pairs. If neither STARTDATE nor STARTTIME is specified, no check is made and the extract begins at the start of the input log file. If ENDDATE and ENDTIME are not specified, the input log tape is processed to end of file. If only STARTDATE is specified, time is not checked; if only STARTTIME is specified, the date is not checked.

If the LOCALTIME parameter has been supplied, the timestamp on each PLOG record is adjusted before the comparison is made to the START and END values. That is, record selection is made based on local time.

Users can specify time intervals in several ways, depending on the results that are desired. The following examples illustrate the most commonly used criteria:

- Select records created on any day within a string of consecutive days. This selects all records created during June 2001.

  ```
  SELECT STARTDATE=20010601 ENDDATE=20010630
  ```

- Select records created after a specified time and before a later time. This selects all records created at or after 10 a.m. on March 3, 2002 and before 4 p.m. on March 10, 2002.

  ```
  SELECT STARTDATE=20020303 STARTTIME=100000
         ENDDATE=20020310   ENDTIME=160000
  ```

- The following selects all records created between 9 a.m. and 5 p.m. for any day on the transaction log. All records with time stamps between 9 a.m. and 5 p.m. is selected, whatever the date of their creation.

  ```
  SELECT STARTTIME=090000 ENDTIME=170000
  ```

- The STARTDATE and ENDDATE can be used to exclude specific days. The following selects all records except for Leap Year day, February 29, 2002.

  ```
  SELECT ENDDATE=20020228  STARTDATE=20020301
  ```

ISN Parameter

```
ISN=H'hex_value'
```

The ISN parameter specifies the Internal Sequence Number. Records associated with the number specified are extracted. The ISN number is specified in a full word (8 digits) hexadecimal format.

```
ISN=H'00C54F01'
```

TSN Parameter

```
TSN=H'hex_value'
```

The TSN parameter specifies the transaction sequence number. Records associated with the number specified are extracted. The TSN number is specified in a full word (8 digits) hexadecimal format.

```
TSN=H'00000601'
```

SELECT: Global Selection of Extracted Records

Syntax Format

The optional SELECT statement takes the following syntax.

```
SELECT
{ [ [EX]GENERAL-ID        = (id_1[,…,id_n])           ]
  [ [EX]GENERAL-ID-CPU-ID = (cpuid_1[,…,cpuid_n])     ]
  [ [EX]GENERAL-ID-OS-ID  = (osid_1[,…,osid_n])       ]
  [ [EX]GENERAL-ID-USERID = (userid_1[,…,userid_n]) ]
  [ [EX]GENERAL-ID-VM-ID  = (vmid_1[,…,vmid_n])       ]
  [ [EX]TERMID            = (term_1[,…,term_n])       ]
  [ [EX]USERID            = (user_1[,…,user_n])       ]
  [ ISN = H'hex_value' ]
  [ STARTDATE = yyyymmdd ENDDATE = yyyymmdd ]
  [ STARTTIME = hhmmss ENDTIME = hhmmss ]
  [ TSN = H'hex_value' ]
};
```

Syntax Example

Each execution of Unicenter CA-PLEU can contain a single SELECT statement. This has the effect of setting global criteria for all the EXTRACT statements that follow. The SELECT statement must follow the GLOBALS statement, if one is present, and precede all EXTRACT statements.

```
SELECT STARTDATE=20010812 STARTTIME=163000
       ENDTIME=183000
       TERMID='TP1','TP3','REM41';
```

How Selection Criteria Are Used

The selection criteria on the SELECT and EXTRACT statements determine which records are extracted from the Protection Log for each output data set. When a record is read from the Protection Log, its contents are compared with the selection criteria specified on the SELECT statement, if one is present.

If the record fails the criteria specified in the SELECT statement, it is discarded, regardless of whatever is specified in the EXTRACT statement.

If the record meets the SELECT statement criteria, its contents are compared with the selection criteria specified on any EXTRACT statement for the Adabas file that this record belongs to.

If it meets those criteria, it is processed as requested on the EXTRACT statement. If it does not, it is discarded.

Records are extracted only if their contents satisfy the selection criteria on both the SELECT and the EXTRACT statements.

Because the SELECT and the EXTRACT statement use identical selection parameters and those parameters are explained in the section on the EXTRACT statement, the parameters available are only listed above. Refer to the EXTRACT statement for their explanation.

Chapter 6

SAVEFDTS Program

This chapter describes how to use the SAVEFDTS program. This program was developed because a Field Description Table (FDT) changes whenever the definition of the fields in the file are changed. For that reason, the current FDT for a file may not give an accurate description of the fields on a Protection Log created some time ago.

SAVEFDTS allows the user to copy the FDTs stored within the Adabas database into a sequential file so that they can be used in the future to process the Protection Log records currently being produced. The use of this sequential file is detailed in the paragraphs describing the FDTDDNAME parameter on the GLOBALS statement.

SAVEFDTS Overview

SAVEFDTS reads the current FDT information via calls to Adabas. The FDT information for Adabas file numbers 1 through 65535 is written into the data set identified by ddname SYSUT1 in IBM z/OS and OS/390, Fujitsu MSP, and Hitachi VOS3 or filename SYS011 or SYS021 in VSE/ESA depending on whether outputting to tape or disk. The contents of this data set are summarized in the printer file defined by ddname SYSPRINT in IBM z/OS and OS/390, Fujitsu MSP, and Hitachi VOS3; SYSLST in VSE/ESA.

Older releases of Adabas require a password to read the FDT for a password protected file. This facility is retained for compatibility. Passwords necessary for reading password protected Adabas file definitions can be specified in the input data set.

SAVEFDTS can be executed in multi-user mode or in single user mode. The execution of SAVEFDTS in multi-user mode has no noticeable effect on database performance.

Input Data Sets

There are two input data sets:

- The file specifying MAXFILES and Adabas passwords.
  - z/OS and OS/390, MSP, and VOS3 ddname: SYSIN
  - VSE/ESA unit: SYSIPT

  The input cards are formatted as follows:

  MAXFILES = {<u>4096</u> | *nnnnn*}

  PASSWORD = {*password* | *password*[,FILE =‘*file_nmbr*’] }

  a. Input parameters are separated by a comma. Reading of parameters is terminated by the first blank. Comments may be included on cards following the first blank.

  b. The MAXFILES parameter is optional. If omitted, it defaults to 4096. When specified, the MAXFILES parameter limits the number of FDTS that are read.

  c. A PASSWORD parameter is necessary only when the files are password protected. It specifies the read password for an individual file or a default read password for all files. This password must conform to Adabas rules for passwords (8 or fewer characters). If a FILE parameter is not specified, this password is used as the default password for reading all file definitions without a specifically supplied password.

  d. The FILE parameter specifies that the preceding PASSWORD parameter on this card is to be used when reading the file definition for the Adabas file specified.

  e. If both MAXFILES and PASSWORD statements are included, the MAXFILES statement must appear first.

  f. Any text following the first blank on the card is taken as a comment and ignored by SAVEFDTS.

- The ADARUN control cards necessary for executing a standard Adabas application program:
  - z/OS and OS/390, MSP, and VOS3 ddname: DDCARD
  - VSE/ESA unit: SYS000, file name CARD

Output Data Sets

There are two output data sets:

- The first data set is the standard print output, containing messages and processing statistics:
  - z/OS and OS/390, MSP, and VOS3 ddname: SYSPRINT
  - VSE/ESA unit: SYSLST

  This data set has a record format of FBA, a logical record length of 133 and a blocksize of 1330. The first byte of each record is the ANSI carriage control character.

- The second output data set contains the file descriptions read from Adabas. This is a sequential data set with variable length, blocked records. The blocksize defaults to 32760, but can be overridden by specifying it in the JCL in the VSE/ESA control card.
  - z/OS and OS/390, MSP, and VOS3 ddname: SYSUT1.
  - VSE/ESA unit: SYS011 or SYS021, file name SYSnnn. SYS011 is used for a tape data set and SYS021 is used for a disk data set.

Considerations When Using SAVEFDTS

Logic

It is necessary for each installation to establish procedures that coordinates the Protection Log files with the appropriate SAVEFDTS output file. These procedures should allow the installation to use logic similar to the following:

- Whenever a file definition is changed, execute SAVEFDTS, keeping track of its date and time of execution.
- For all Protection Logs created since the last file definition change, run Unicenter CA-PLEU using either the last SAVEFDTS file or the online Adabas file definitions.
- For all Protection Logs created prior to the last file definition, use the appropriate SAVEFDTS output file.

The SAVEFDTS output file contains the date and time when the field definitions were read from Adabas. In addition, Unicenter CA-PLEU contains logic that checks the date and time of the Protection Log record against the date and time of the field definition. A warning message is issued if the Protection Log record is older than the field definition. This prevents users from being unaware that the wrong field definition was used to process a Protection Log.

Adabas Availability

A copy of the current field definitions enables Unicenter CA-PLEU to execute without requiring Adabas to be operational. Those installations that keep Adabas in operation during just a portion of the day can delay processing of the Protection Log without having to activate Adabas just to process the log.

The SAVEFDTS output file also permits an installation running multiple Adabas databases, whether on the same or different machines, to process the Protection Logs without regard for availability of the Adabas database itself.

Access Authorization

When executing SAVEFDTS, it is critical that you supply an Adabas password which has an access authorization level high enough to allow access to any field in any file where FDTs are read. Otherwise, any fields not available to the job is deleted from the information returned to SAVEFDTS by the LF commands, and the resulting FDT information will not match data on Protection Logs.

Note: Adabas gives no indication that field definitions have been withheld by the LF command.

Subsequent attempts to process Protection Logs using the incomplete SAVEFDTS information could result in abnormal termination of Unicenter CA-PLEU, and, if Unicenter CA-PLEU output files are used to update the database, corrupted data in the database.

Execution in z/OS and OS/390, MSP, and VOS3

Sample JCL

The SAVEFDTS program is executed as a standard job or job step.

```
JOBCARD1
JOBCARD2
//STEP1    EXEC PGM=SAVEFDTS
//STEPLIB  DD   DSN=hlq.mlq.PLEUVxxx.LOAD,DISP=SHR
//         DD   DSN=ADABAS.LOADLIB,DISP=SHR
//DDCARD   DD   DSN=ADABAS.CNTL,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
MAXFILES=255
PASSWORD=HYANDMYT
PASSWORD=SECRET,FILE=23
PASSWORD=PUBLIC,FILE=7
/*
//SYSUT1   DD   DSN=hlq.mlq.SAVEFDTS.OUTPUT,DISP=(,NEW,CATLG),
//              UNIT=DISKUNIT,SPACE=(TRK,(15,5))
/*
```

Job Control Language Explanation

The following explanations refer to the JCL above.

- The first //STEPLIB is required. It identifies the load library that contains the SAVEFDTS program.
- The second //STEPLIB card identifies the load library containing ADARUN and other Adabas execution modules.
- //DDCARD is required. It identifies the ADARUN control cards appropriate for running a batch Adabas program at your installation. SAVEFDTS uses standard Adabas calls to read the Adabas Field Description Tables.
- //SYSPRINT is required. It identifies the output data set where SAVEFDTS writes its messages and processing summary.
- //SYSIN is required. It contains control cards specifying the maximum number of FDTS to be read, and any passwords necessary for read access to the Adabas files. They must begin in column 1. If the default MAXFILES is used and no Adabas files are security protected, SYSIN may be specified as DUMMY, or be an empty data set. If both MAXFILES and PASSWORD statements are included, the MAXFILES statement must appear first.
- //SYSUT1 is required. It identifies the sequential data set where SAVEFDTS writes the Field Description Tables that are read from Adabas.

Execution in VSE/ESA

Sample JCL

The SAVEFDTS program is executed as a standard job.

```
// JOB     JOBNAME YOUR INSTALLATION'S ACCOUNTING INFO
// LIBDEF  PHASE,SEARCH=(yourlib.PLEUVxxx,ADALIB.ADAVxxx),TEMP
// ASSGN   SYS000,DISK,VOL=vvvvvv,SHR
// DLBL    CARD,'ADABAS.ADARUN',0,SD
// EXTENT  SYS000,vvvvvv,...
// ASSGN   SYS011,TAPE
// TLBL    SYS011,'SAVEFDTS.OUTPUT'
// EXEC    SAVEFDTS
MAXFILES=255
PASSWORD=HYANDMYT
PASSWORD=SECRET,FILE=23
PASSWORD=PUBLIC,FILE=8
/*
/&
```

Job Control Language Explanation

The following explanations refer to the JCL above.

- // JOB is required. It should be coded according to your installation's job accounting conventions.
- // LIBDEF is optional. The LIBDEF statement identifies the library that contains the SAVEFDTS phase. The LIBDEF statement may also identify the library containing ADARUN and other modules necessary for Adabas execution.
- // ASSGN SYS000 and associated CARD DLBL and EXTENT cards are required. They identify the ADARUN control cards appropriate for running a batch Adabas program at your installation. If ADARUN is set up for appropriate defaults, SYS000 may be set to IGN. SAVEFDTS uses standard Adabas calls to read the FDTs.
- // ASSGN SYS011 identifies the output data set from the program as residing on tape. If output to disk is desired, the name SYS021 should be used. Only one output data set can be written in a single execution of SAVEFDTS, so either SYS011 or SYS021 must be specified, but not both.
- // EXEC is required. It identifies the program to be executed.
- Cards following the EXEC card are the SYSIPT input specifying the maximum number of FDTS to be read, and any passwords necessary for read access to the Adabas files. They must begin in column 1. If the default MAXFILES is used and no Adabas files are security protected, these cards may be omitted. If both MAXFILES and PASSWORD statements are included, the MAXFILES statement must appear first.
- All printed output is written to SYSLST.

Condition Codes at Termination

Code Explanation

The following condition codes are returned by the program upon termination:

Code	Description
0	The file descriptions for all Adabas files were written to the output data set and the program terminated normally.
16	An error has occurred which prevented creation of the output data set containing the field definitions; the nature of the error is described by an error message written to the printer.

Chapter
7 User Exits

Some installations want to employ record selection criteria beyond those provided by Unicenter CA-PLEU Protection Log Extract Utility for Adabas (Unicenter CA-PLEU). Other installations may want to perform some pre-processing of selected records before they are written to the extract file. This may include accumulating summary statistics, inserting new records into the extract file or modifying the Protection Log record before it is written. Still other installations wish to keep selected Protection Log records online within their Adabas database.

To support these requirements, Unicenter CA-PLEU can call user-written exit routines prior to writing records to the extract data sets. The purpose of this chapter is to define the calling conventions used and to describe ways exit routines can be used.

Available User Exit Functions

The user exit is presented with the formatted Protection Log record and other information necessary to process it. The following list summarizes some actions which can be taken by the user exit. The user exit can:

- Perform additional selection testing on the record
- Extract data from the record for the purpose of accumulating summary statistics
- Modify the record passed to it, perhaps altering the record length
- Generate records that are written into the extract data set either in addition to or instead of the Protection Log record
- Access Adabas, either through direct calls or through Adabas SQL Server (AQA)
- Read or write data to or from data sets which it opens and closes
- Stop processing of the EXTRACT request

Calling Routine Functions

The routine which calls the user exit performs further processing on the record as instructed by the return code from the user exit. The available options are:

- Write the record, which may have been modified by the user exit, to the extract data set
- Delete the record, that is, do not write the record to the extract data set
- Insert a record built by the user exit and call the exit routine again to complete processing of the present Protection Log record

The routine which calls the user exit responds to a request for inserting a record by writing the inserted record to the extract file and calling the user exit again with the same Protection Log record, which might have been modified by the user exit. The user exit can then insert another record or choose to write out the Protection Log record or delete the Protection Log record.

When coding a user exit that inserts records, remember that the user exit is called repeatedly for each Protection Log record until a return code is provided that results in either writing the Protection Log record out or deleting it. Therefore, the user exit needs to contain logic allowing it to differentiate between the first, second, etc. calls for a given Protection Log record.

Calling Sequence

The calling sequence for the user exit uses standard linkage conventions. The exit routines can be written in any language, including Assembler and COBOL, that adheres to those conventions. Sample exit routines that demonstrate the linkage are included later in this chapter.

Register Contents

On entry to the user exit, the general registers contain the following information. In this discussion, R0 refers to register 0, R1 to register 1, etc.

Register	Contents
R0	Unpredictable
R1	The address of the parameter list described below
R2 through R12	Unpredictable
R13	The address of an 18-word save area

Register	Contents
R14	The address to which the exit should return control
R15	Entry point of the exit routine

The exit routine must preserve the values of R2 through R13 and restore them to their values at entry to the routine before exiting. The save area pointed to by R13 at entry to the exit routine can be used to save the register contents.

Parameter List

On entry to the exit routine R1 points to a parameter list containing eight full-word addresses; the use of each parameter is described in detail below. The parameter list contains the following addresses:

- Address of a full word for storing the return code.
- Address of the formatted Protection Log record.
- Address of a full word containing the length of the record buffer.
- Address of the format buffer supplied by the user in the EXTRACT statement or of the format buffer built by Unicenter CA-PLEU if the ADDFORM or ADDFORM-LONG formats was requested. This value is undefined if UNLOAD format was requested.
- Address of a half word that contains the Adabas file number of the Protection Log record.
- Address of an eight-character area that contains the data set name of the file to which this record is written.
- A full word user area.
- Address of a full word containing the length of the format buffer pointed to by parameter 4.

The contents and use of each of these parameters are described in detail in the following sections.

Address of a Full Word Return Code

This points to a full word where the exit routine should store its return code for the calling program. Here are the meanings of return codes, given in decimal values:

Return Code	Description
-1	Stop processing of the EXTRACT request. Unicenter CA-PLEU recognizes a user-exit return code value of -1 (a full word of hex 'FF's) as a signal to stop processing of the EXTRACT request for which the user-exit was called. When the -1 return code is detected, the current record in the record buffer is written and no further processing is done for that extract. If all EXTRACT requests in a given execution of Unicenter CA-PLEU have been terminated in this manner, Unicenter CA-PLEU terminates normally as if end-of-file had been reached on the Protection Log. This option allows reading of the Protection Log to be stopped when no additional information is desired.
0	Write out record as it appears in the record buffer.
8	Skip the present Protection Log record.
16	Insert a record. That is, write out the contents of the record buffer and call the exit routine again before reading another Protection Log record.

Note: It is the contents of this full word, rather than the contents of R15, that is set by the exit. This full word must be set to zero (0) upon return from the exit routine if a 0 return code, that is, writing out the record, is desired.

Address of the Formatted Record

This value contains the address of a buffer area. This buffer contains the formatted record that is ready for output to the extract data set. This record is in variable length format, meaning that the first half word contains the record length and the second half word is zero. If the user exit modifies the record length, the new length must be inserted into the first half word of the record.

Records to be inserted must be built in this area. They must also be in variable length format, with their length in the first half word and zeroes in the next half word. If the Protection Log record is written out following the record insertion, the user exit must save the Protection Log record in its own working storage so the record can be moved back to this area for output.

The user exit is called after reaching end of file on the Protection Log. At this time additional records can be inserted before the output data set is closed. End of file on the Protection Log is indicated to the user exit by presenting a record whose length has been set to X'FFFF'. In COBOL one should test for -1.

As many records as desired can be inserted by building them in the record area provided, overlaying the X'FFFF' record length with the actual record length. As with any other record insertion, the exit routine is called repeatedly until a return code to delete or write out the Protection Log record is returned. Since in this case there is no Protection Log record to write or to delete, using either of these return codes terminates the insertion of records for this extract.

Address of the Length of the Record Buffer	This value is the address of a full word that contains the length of the buffer area for building output records. The contents of that word must not be changed by the user exit. The address in the previous parameter is the start of this buffer area. When coding a user exit routine that builds records for insertion or extends formatted records, one must avoid placing information beyond the end of this buffer; doing so overlays control block information and probably cause Unicenter CA-PLEU to malfunction.
Address of the Format Buffer	This address points to the format buffer supplied by the user in the EXTRACT statement which resulted in this call to the exit; or, if the ADDFORM or ADDFORM-LONG format was requested, the format buffer is built by Unicenter CA-PLEU. This address is unpredictable if the format specification is UNLOAD. The format buffer must not be changed by the user exit.
Address of the Adabas File Number	This value is the address of a half word. The half word contains the binary value of the Adabas file number of the Protection Log record presently in the record buffer. This value helps exit routines which are called for more than one Adabas file to identify what type of processing is required. This is particularly true at end-of-file on the Protection Log when there is no log record to test. The exit routine must not alter the value of the file number.
Address of Data Set Name	This value is the address of an eight-byte area that contains the data set name of the file that receives the output from the extract. The value of the data set name must not be altered by the user exit.
User Area	This full word is available for use by the user exit. It is initialized to zero and any value placed in it by the exit routine is preserved between calls to the user exit. An assembler language exit routine can use this area to store the address of a work area that is obtained on the first call to the exit routine. In this way, assembler language exit routines can be made re-entrant.
Length of the Format Buffer	This full word points to a full word binary number which is the length of the format buffer pointed to by the fourth parameter. It must be used in processing ADDFORM output. The length of the format buffer varies from record to record, depending on the presence or absence of null fields. This value can be used in the format-buffer length of an Adabas Control Block if direct Adabas processing is performed in the exit.

Parameter Usage

The addresses of the Adabas file number and the data set name are provided for user exit routines called by more than one extract. This enables them to differentiate between the source of the calls and the types of records that need to be processed.

Record Area Modifications

The user exit may modify the contents of the record area pointed to by the second parameter and it may also modify the contents of the user area pointed to by the seventh parameter. The user exit must not modify the contents of any other area pointed to by this parameter list.

Miscellaneous User Exit Considerations

Multiple Exit Routines

Each EXTRACT statement can identify a user exit. It is valid for more than one EXTRACT statement to invoke a single exit routine. This can be done if the purpose of the exit routine is to perform additional selection testing on several of the Adabas files being processed. Remember that only one copy of each exit routine is loaded, regardless of how many EXTRACT statements may refer to it. This must be kept in mind when designing the initialization, processing and termination routines for the exit.

Also remember that exit routines with different names are independent of each other. If multiple EXTRACT statements are provided for a single Adabas file number and if each of those statements identifies a unique user exit, the processing of each exit routine does not affect the actions of the other exits.

For example, a decision by one user exit to skip a Protection Log record affects only that extract. The record is formatted for subsequent EXTRACT statements and presented to the user exits which they call. Likewise, data modifications made by one user exit does not alter the contents of records formatted for other EXTRACT statements.

Accessing Adabas

User exits may access Adabas, either through direct calls or through Adabas SQL Server (AQA). In this way, data from the Protection Log can be written directly into its own Adabas files. Summary or exception records that are written into an Adabas file to provide online summary and exception reporting can be generated.

Accessing Other Data Sets

User exits may access data sets, either for input or output. The selected ddnames, link names, or filenames and logical unit numbers must not conflict with those used by Unicenter CA-PLEU or referred to by control statements.

The appropriate job control statements must be added to the JCL described in the preceding examples. The exit routine is responsible for verifying that the data sets are properly opened and closed.

Abnormal Terminations

Because the user exit has access to all of the working storage used by Unicenter CA-PLEU, it is possible for errors in an exit routine to destroy information required for correct execution of Unicenter CA-PLEU. When this happens, the errors that occur can sometimes appear to be unrelated to the actions of the user exit. Therefore, when a malfunction occurs and user exits are being used, please attempt to recreate the problem after removing the user exits. If the problems persist, diagnosis is simplified. If the problems go away, further analysis of the actions of the user exits is warranted.

Sample COBOL Exit

The following pages contain a sample user exit routine written in COBOL. This sample is provided to show how the linkage should be established between Unicenter CA-PLEU and the user exit.

This program is called in response to the presence of the EXITNAME keyword with a value of COBEXIT on one or more EXTRACT statements.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.   COBEXIT.
REMARKS.      DEMONSTRATION COBOL USER EXIT FOR UNICENTER CA-PLEU
              PROTECTION LOG EXTRACT UTILITY FOR ADABAS.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
*
*    THE FOLLOWING SAVE-RECORD AREA AND SWITCH WILL BE USED IF IT
*    IS NECESSARY TO SAVE RECORDS WHEN RECORDS ARE BEING INSERTED.
*
 01   SAVE-RECORD         PIC X(...).
 01   INSERT-SWITCH       PIC X VALUE SPACE.
      88 RECORD-INSERTED        VALUE 'I'.
*
LINKAGE SECTION.
*
*    THE USER SETS RETURN-CD TO INDICATE ACTION DESIRED
*      0 -- WRITE OUT THE RECORD
*      8 -- DO NOT WRITE OUT THE RECORD (DELETE
*     16 -- WRITE THE RECORD IN RECORD BUFFER AND
*           RETURN IMMEDIATELY (INSERT A RECORD).
*
 01    RETURN-CD          PIC S9(7) COMP.
*    DECOMPRESSED RECORD CONTAINS THE RECORD THAT WILL BE WRITTEN
*    TO THE OUTPUT FILE. IT CONSISTS OF THE STANDARD EXTRACT HEADER
*    AND THE RECORD IN THE FORMAT REQUESTED IN THE EXTRACT
*    STATEMENT.
*
 01    DECOMPRESSED-RECORD.
*    RECORD-HEADER IS THE STANDARD EXTRACT
*    RECORD HEADER FROM THE EXTRACT UTILITY.
*
 02   RECORD-HEADER.
*
*   RECORD-LENGTH IS ORDINARILY THE
*   LENGTH OF THE DECOMPRESSED RECORD.
*   IT WILL BE SET TO -1 AT THE
*   END-OF-FILE CALL OF THE EXIT.
*
 04 RECORD-LENGTH        PIC S9(4) COMP.
 04 FILLER               PIC X(2).
 04 USER-ID              PIC X(8).
 04 TERMINAL-ID          PIC X(4).
 04 FILE-NUMBER          PIC S9(4) COMP.
 04 RECORD-TYPE          PIC X(2).
 04 RECORD-ISN           PIC S9(7) COMP.
 04 RECORD-DATE          PIC 9(8).
 04 RECORD-TIME          PIC 9(6).
 04 DATABASE-NAME        PIC X(8).
 04 DATABASE-ID          PIC S9(4) COMP.
 04 CPU-ID               PIC X(8).
 04 DATA-LENGTH          PIC S9(4) COMP.
 04 FILLER               PIC X(2).
 04 TRAN-SEQ-NUM         PIC S9(8) COMP.
 04 CHKPT-NAME           PIC X(4).
 04 RABN                 PIC S9(8) COMP.
 04 CLUSTER-NAME         PIC S9(4) COMP.
 04 PRTY                 PIC S9(4) COMP.
 04 COMMAND-COUNT        PIC S9(8) COMP.
 04 BLOCK-COUNT          PIC S9(8) COMP.
 04 RECORD-NUM           PIC S9(4) COMP.
 04 SESSION-NUM          PIC S9(4) COMP.
 04 UNIQUE-ID.
    05 UNIQ-CPUID        PIC X(8).
    05 UNIQ-VMID         PIC X(8).
    05 UNIQ-OSID         PIC X(4).
    05 UNIQ-USERID       PIC X(8).
 04 WORK-RABN-CHAIN      PIC X(2).
 04 FILLER               PIC X(10).
*
*
*  THE FOLLOWING CONTAINS THE RECORD IN THE FORMAT REQUESTED BY
*  THE FORMAT BUFFER IN THE EXTRACT STATEMENT. IT MAY BE
*  MODIFIED BY THE USER AS NECESSARY, OR DATA SUBSTITUTED IN
*  ORDER TO PERFORM AN INSERT.
*
 02   RECORD-CONTENTS.
 04   FIELD1              PIC...
 04   FIELD2              PIC...
*
*  BUFFER-LENGTH CONTAINS THE LENGTH OF THE OUTPUT BUFFER
*  ADDRESSED BY THE ABOVE 01-LEVEL. INSERTED RECORDS MUST
*  NOT BE LONGER THAN BUFFER-LENGTH.
*
 01   BUFFER-LENGTH       PIC S9(7) COMP.
*
* FORMAT-BUFFER CONTAINS THE FORMAT BUFFER CHARACTER
* STRING PROVIDED IN THE EXTRACT STATEMENT.
*
 01   FORMAT-BUFFER       PIC X(..).
*
*  ADABAS-FILE-NUMBER CONTAINS THE FILE-NUMBER FOR THE EXTRACT
*  STATEMENT. THIS WILL ORDINARILY BE THE SAME AS IN THE RECORD
*  HEADER, BUT MAY BE HELPFUL AT THE END OF FILE CALL.
*
 01   ADABAS-FILE-NUMBER  PIC S9(4) COMP.
*
*  OUTPUT-DDNAME IS THE DD NAME TO WHICH THIS EXTRACT
*  IS BEING WRITTEN. DO NOT MODIFY!
*
 01   OUTPUT-DDNAME       PIC X(8).
*
*  DUMMY IS A DUMMY PARAMETER; NOT USEABLE IN COBOL
```

```
*
 01   DUMMY               PIC S9(7) COMP.
*
*  FORMAT-BUFFER-LENGTH IS THE LENGTH OF THE FORMAT BUFFER
*  SUPPLIED BY THE USER, OR BUILT BY THE UTILITY FOR
*  FORMAT=ADDFORM
*
 01   FORMAT-BUFFER-LENGTH PIC S9(7) COMP.
*
*
PROCEDURE DIVISION USING
       RETURN-CD, DECOMPRESSED-RECORD,
          BUFFER-LENGTH, FORMAT-BUFFER,
          ADABAS-FILE-NUMBER, OUTPUT-DDNAME,
          DUMMY, FORMAT-BUFFER-LENGTH
*
    IF RECORD-LENGTH = -1
       PERFORM 900-END-OF-EXTRACT THRU
               999-END-OF-EXTRACT-EXIT
       GOBACK.
*
    IF...
       PERFORM 100-CHANGE-RECORD THRU
               199-CHANGE-RECORD-EXIT
       GOBACK.
*
*
    IF...
       PERFORM 200-DELETE-RECORD THRU
               299-DELETE-RECORD-EXIT
       GOBACK.
*
    IF...
       PERFORM 300-INSERT-RECORD THRU
               399-INSERT-RECORD-EXIT
       GOBACK.
*
    IF RECORD-INSERTED
       PERFORM 400-AFTER-INSERT THRU
               499-AFTER-INSERT-EXIT
       GOBACK.
*
    ELSE
       MOVE 0 TO RETURN-CD
       GOBACK.
*
100-CHANGE-RECORD.
    MOVE... TO FIELD1.
    MOVE 0 TO RETURN-CD.
199-CHANGE-RECORD-EXIT.
    EXIT.
*
200-DELETE-RECORD.
    MOVE 8 TO RETURN-CD.
299-DELETE-RECORD-EXIT.
    EXIT.
*
300-INSERT-RECORD.
    MOVE DECOMPRESSED-RECORD TO SAVE-RECORD
    MOVE 'I' TO INSERT-SWITCH.
    MOVE... TO DECOMPRESSED-RECORD.
    MOVE 16 TO RETURN-CD.
399-INSERT-RECORD-EXIT.
    EXIT.
```

```
*
*  PUT OUT A SAVED RECORD FOLLOWING AN INSERT.
400-AFTER-INSERT.
    MOVE SPACE TO INSERT-SWITCH.
    MOVE SAVE-RECORD TO DECOMPRESSED-RECORD.
    MOVE 0 TO RETURN-CD.
499-AFTER-INSERT-EXIT.
    EXIT.
```

Sample Assembler Exit

The following pages contain the skeleton of a user exit written in Assembler. The code shows how the parameter list is used and how linkage is established. It also shows how the user area of the parameter list can be used to create a reentrant exit routine.

Users of VSE/ESA operating systems would have to change the GETMAIN macro to the analogous GETVIS macro.

Sample assembler exit is shown below:

```
USEREXIT   CSECT
           REQU                               EQUATES REGISTERS.
           STM         R14,R12,12(R13)        SAVE REGISTERS.
           LR          R12,R15                ENTRY POINT.
           USING       USEREXIT,R12
           LR          R2,R1                  PARAMETER LIST ADDR.
           USING       PARMLIST,R2
*
*  SEE IF A WORK AREA WAS GOTTEN PREVIOUSLY.
*  IF NOT, GET IT NOW.
*  NOTICE THAT THIS ROUTINE IS REENTRANT.
*
           L           R1,PARMUSER             WORK AREA ADDR.
           LTR         R1,R1                   GOT ONE YET?
           BNZ         START                   BR IF SO.
           LA          R0,WORKLNTH             LENGTH OF AREA.
           GETMAIN R,LV=(0)
           ST          R1,PARMUSER             INTO PARM LIST.
 START     ST          R1,8(,R13)              INTO PREV S/A.
           ST          R13,4(,R1)              PREV S/A ADDR.
           LR          R13,R1                  RESET SAVE AREA PTR.
           USING       WORKAREA,R13
           LM          R3,R8,PARMLIST          ADDRS FROM PARM LIST.
           ST          R3,WORKRETA             ADDR FOR RET CODE.
           USING       RECORD,R4
           L           R5,0(,R5)               BUFFER LENGTH.
           ST          R6,WORKUFBA             USER FMT BUFFER ADDR.
           LH          R7,0(,R7)               Adabas FILE NO.
           ST          R8,WORKDDNA             DDNAME ADDR.
           DROP        R2
           CH          R5,=H'1'                END OF LOG TAPE?
           BNH         ENDOFTAP                BR IF SO.
```

```
*
*  CODE TO PROCESS PROTECTION LOG RECORDS WOULD
*  BE INSERTED HERE.
*
ENDOFTAP EQU *
*
*  CODE TO COMPLETE EXIT PROCESSING AFTER LAST
*  PROTECTION LOG RECORD GOES HERE.
*
*  THE FOLLOWING CODE SETS THE RETURN CODES TO
*  DELETE THE FORMATTED RECORD (SKIP WRITING
*  IT OUT), TO INSERT A RECORD OR TO WRITE
*  THE FORMATTED RECORD TO THE EXTRACT FILE.
*
DELETE    LA     R2,8            SKIP THIS RECORD.
          B      RETURN          GO RETURN.
INSERT    LA     R2,16           RECORD INSERTED.
          B      RETURN
WRITEREC  SR     R2,R2           WRITE RECORD.
RETURN    L      R3,WORKRETA     ADDR FOR RET CODE.
          ST     R2,0(,R3)       PASS RET CODE.
          L      R13,4(,R13)     PREV S/A.
          LM     R14,R12,12(R13) RESTORE REGS.
          BR     R14             RETURN.
          LTORG
          DROP   R4,R12,R13
*
*  THE FOLLOWING DSECT MAPS THE FORMATTED
*  PROTECTION LOG RECORD.
*
RECORD    DSECT
RECDLNTH  DS     H     RECORD LENGTH
          DS     H     UNUSED.
RECDUSER  DS     CL8   USER ID.
RECDTERM  DS     CL4   TERMINAL ID.
RECDFINO  DS     H     Adabas FILE NO.
RECDTYPE  DS     CL2   FILE TYPE.
RECDAFTR  EQU    C'A'  AFTER IMAGE.
RECDBEFR  EQU    C'B'  BEFORE IMAGE.
RECDISN   DS     XL4   ISN.
RECDDATE  DS     CL8   DATE OF RECD
RECDTIME  DS     CL6   TIME OF RECD
RECDDBNM  DS     CL8   DBNAME FROM GLOBALS.
RECDDBID  DS     H     DB ID.
RECDCPID  DS     CL8   CPU ID FROM GLOBALS.
RECDDATL  DS     H     LENGTH OF OUT5DATA AREA IN BYTES.
          DS     H     FULL-WORD ALIGNMENT SLACK BYTES.
RECDTSN   DS     F     TRANACTION SEQUENCE NUMBER.
RECDCPNM  DS     CL4   CHECK POINT NAME.
RECDRABN  DS     F     RABN.
RECDCLUS  DS     H     CLUSTER NUMBER (MAYBE).
RECDPRTY  DS     H     PRIORITY.
RECDCMDC  DS     F     COMMAND COUNT.
RECDBLCT  DS     F     BLOCK COUNT FROM PLOG TABLE.
RECDRCNO  DS     H     RECORD NUMBER WITHIN BLOCK.
RECDSESS  DS     H     Adabas SESSION NUMBER.
RECDUNIQ  DS     0CL28 GENERAL UNIQUE IDENTIFIER.
RECDUNCP  DS     XL8   GENERAL UNIQUE IDENTIFIER, CPU-ID.
RECDUNVM  DS     CL8   GENERAL UNIQUE IDENTIFIER, VM-ID.
RECDUNOS  DS     XL4   GENERAL UNIQUE IDENTIFIER, OS-ID.
RECDUNID  DS     XL8   GENERAL UNIQUE IDENTIFIER, USERID.
*
RECDWRBN  DS     XL2   WORK-RABN-CHAIN.
          DS     XL10  RESERVED.
RECDHDRL  EQU    *-RECORD HEADER-LENGTH.
RECDDATA  EQU    *     FIRST DATA BYTE.
```

```
*
*  THE FOLLOWING DSECT MAPS THE PARAMETER LIST.
*
PARMLIST  DSECT
PARMRETA  DS     A     ADDR FOR RETURN CODE.
PARMRECA  DS     A     FORMATTED RECD ADDR.
PARMLENA  DS     A     ADDR OF BUFFER LENGTH.
PARMUFBA  DS     A     USER FORMAT BFFR ADDR
PARMFINA  DS     A     ADABAS FILE NO. ADDR
PARMDDNA  DS     A     DDNAME ADDRESS.
PARMUSER  DS     F     USER AREA.
PARMFBLN  DS     A     ADDRESS OF FORMAT BUFF LEN
*
*  THE FOLLOWING DSECT MAPS THE WORK AREA.
*
WORKAREA  DSECT
WORKSAVE  DS     18F        OS SAVE AREA.
WORKRETA  DS     A          ADDR. FOR RET CODE.
WORKUFBA  DS     A          USER FMT BUFFER ADDR
WORKDDNA  DS     A          DDNAME ADDR.
          DS     0D         DOUBLE WORD BOUNDARY.
WORKLNTH  EQU    *-WORKAREA LENGTH OF AREA.
          END    USEREXIT
```

Chapter 8 Unicenter CA-PLEU Messages

The following messages are issued by Unicenter CA-PLEU Protection Log Extract Utility for Adabas (Unicenter CA-PLEU). These messages are all preceded by the three-character prefix DAT whenever they are printed. Messages that do not have the DAT prefix are not issued by Unicenter CA-PLEU.

Unicenter CA-PLEU Messages

DAT00001I CONTROL CARD INPUT:

Explanation	The control card input to the program is listed following this message. Processing continues.
Action	None required.

DAT00002I SYNTAX PROCESSING SUCCESSFULLY COMPLETED

Explanation	Syntax checking of you-supplied control cards has been completed and no syntax errors were found. Processing continues.
Action	None required.

DAT00003I OUTPUT REQUEST ANALYSIS SUCCESSFULLY COMPLETED

Explanation	Analysis of the requests for output data sets (extract files) has been completed without error. Processing continues.
Action	None required.

DAT00004I UNICENTER CA-PLEU HAS COMPLETED WITH RETURN CODE *n*

Explanation	Unicenter CA-PLEU has completed with the indicated return code. Processing terminates.
Action	A return code of zero (0) indicates satisfactory completion. Any other return code follows one or more messages that indicate an unusual condition. The return code will be equal to the highest severity of those messages. You should examine these messages and respond accordingly.

DAT01001E IDENTIFIER IS LONGER THAN 32 CHARACTERS

Explanation	Keyword values longer than 32 characters are not allowed. The message indicates the location of the error in the control statement. Program execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT01002E STRING IS LONGER THAN 127 CHARACTERS

Explanation	The maximum allowable length of a string is 127 characters. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT01003E STRING NOT TERMINATED BEFORE END OF LINE

Explanation	Strings cannot be continued past the end of a control statement. Long strings can be generated through concatenation. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT01004E INVALID HEXADECIMAL CHARACTER

Explanation	The indicated character is not a valid hexadecimal value. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT01005E HEX STRING NOT TERMINATED BEFORE END OF LINE

Explanation	Hexadecimal strings may not be continued across control statement boundaries. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT01006E UNEVEN NUMBER OF HEX CHARACTERS

Explanation	Hexadecimal characters must be presented in pairs to make up a complete byte of information. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT01007E CONTINUATION OF STRING NOT FOUND

Explanation	The control statement indicated that a string continuation should be expected, but none was found. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT01010E FIXED NUMBER HAS TOO MANY DIGITS

Explanation	Fixed numbers are limited to 10 digits plus sign. This is sufficient to contain the largest and smallest fixed numbers that can be processed by Adabas. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT01012E AREA REQUESTED LARGER THAN MAXIMUM SIZE *n*

Explanation	This is an internal error, not a user error. A request for virtual storage exceeded the maximum amount available. Processing terminates.
Action	Contact Computer Associates or your local distributor with a description of the error.

DAT01013E READING BACKWARDS FROM DISK NOT PERMITTED

Explanation	The GLOBALS parameter READ-BACWARD=YES was specified while attempting to process Adabas Protection Logs from a disk data set. Reading backwards is valid only when processing Adabas Protection Logs from tape. Processing terminates.
Action	Copy the Adabas Protection Logs to tape if backward processing is required. Otherwise, change the READ-BACKWARDS parameter to NO and resubmit the request.

DAT02001E MISSING STATEMENT IDENTIFIER

Explanation	No statement identifier, SELECT or EXTRACT, was found at the start of a control statement sequence. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02002E INVALID STATEMENT IDENTIFIER

Explanation	This is an internal error, not a user error. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Contact Computer Associates or your local distributor with a description of the error.

DAT02003E INVALID TOKEN IN SELECT STATEMENT

Explanation	One of the tokens encountered while parsing the SELECT statement is not valid. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02004E INVALID PARAMETER FOR SELECT

Explanation	A parameter supplied on a SELECT statement is not a valid parameter. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02005E INVALID PARAMETER NAME

Explanation	The indicated parameter name is not valid. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02006E INVALID PARAMETER IN EXTRACT

Explanation	The indicated parameter is not valid within an EXTRACT control statement. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02007E INVALID PARAMETER FOR GLOBALS

Explanation	A parameter supplied on a GLOBALS statement is not a valid parameter. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02021E INVALID NUMERIC PARAMETER

Explanation	The indicated numeric parameter is not valid. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02022E MISSING MATCHING PAREN ON PARAMETER

Explanation	One or more parentheses have been omitted from the indicated statement. The number of right parentheses must equal the number of left parentheses. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02023E INVALID ALPHABETIC PARAMETER

Explanation	The indicated alphabetic parameter is not valid. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02024E INVALID PARAMETER TYPE

Explanation	The indicated parameter is not a valid type in the context of the control statement on which it was encountered. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02025E INVALID DATE PARAMETER

Explanation	The date parameter must be supplied as eight characters in the format yyyymmdd, where yyyy is the four digits of the year, mm the month and dd the day of the month. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02026E INVALID TIME PARAMETER

Explanation	The time parameter must be supplied as six characters in the format hhmmss, where hh is the hour on a 24-hour clock, mm the minute and ss the second. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02027E NO DD STATEMENT IN JCL FOR FDTDDNAME

Explanation	The file identified by the FDTDDNAME keyword value could not be found in the job control. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02028E INVALID LOCAL TIME ZONE

Explanation	The value spplied in the LOCALTIME parameter does not specify a range of 1-12 hours East or West of GMT.
Action	Correct the error and execute the program again.

DAT02031E INVALID FILE NUMBER, GT MAXFILES

Explanation	The maximum possible Adabas file number is 65535. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02032E DDNAME REQUIRED FOR EXTRACT

Explanation	No DDNAME parameter was specified for this EXTRACT request. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02033E FORMAT BUFFER REQUIRED FOR EXTRACT

Explanation	No FORMAT, BFORMAT, or AFORMAT parameter was specified for this EXTRACT request. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02034E FILE NUMBER REQUIRED FOR EXTRACT

Explanation	No FILE parameter was specified for this EXTRACT request. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02035E NO DDNAME FOR EXTRACT IN JCL

Explanation	The ddname/filename was not defined in the JCL. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT02036E EXIT MODULE NOT FOUND IN STEPLIB

Explanation	You exit routine identified by the EXITNAME keyword parameter could not be found. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT03001E THE ADABAS FILE DEFINITION WAS NOT READ, ADABAS RESPONSE CODE IS *n*

Explanation	The Adabas LF commands to read the file definitions failed with the indicated response code. This is usually caused by either a file number that is not defined in the database (response 017) or Adabas not being active (response 148). Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

Note: The following error messages indicate incorrect Adabas format buffers for the selected file. You should refer to the *Adabas Command Reference Manual* for the syntax of the format buffer specification, and the ADAREP report for the format of fields in the file.

DAT40001E INVALID ADABAS FIELD NAME

Explanation	The field name indicated was not found in the Field Definition Table for this file. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT40002E NO SECOND CHARACTER TO FIELD NAME

Explanation	The format interpreter expected the previous alpha character to be the first character of a two-character Adabas field name, but did not find a second character. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT40003E NO CLOSING PAREN TO MU RANGE

Explanation	An opening parenthesis was found for an MU field, but no closing parenthesis was found. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT40004E EXPECTING , OR .

Explanation	A comma or period was expected at this point in the format buffer. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT40005E NO COMMA FOLLOWING LENGTH

Explanation	A comma was expected at this point in the format buffer. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT40006E INVALID FIELD FORMAT

Explanation	A field format was not A, B, F, G, P, or U. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT40007E NO COMMA AT END OF FIELD ENTRY

Explanation	A comma was expected at this point in the format buffer. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT40010E NO PERIOD IN FORMAT BUFFER

Explanation	No period was found at the end of the format buffer string. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41001E NO LENGTH ALLOWED FOR GROUP FIELD

Explanation	A group field was specified followed by a field length. This is an illegal usage. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41002E NO FORMAT ALLOWED FOR GROUP FIELD

Explanation	A group field was specified followed by a field format. This is an illegal usage. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41003E GROUP FIELD MAY NOT CONTAIN AN MU FIELD

Explanation	A group field was named that contains an MU field. Such a group field name may not be used in a format buffer. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41010E NO SUCH FIELD IN FILE

Explanation	The field name was not found in the Field Definition Table for this Adabas file. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41011E PE OR MU INDEX 0 OR 191 OR NOT SUPPLIED

Explanation	The index supplied for an MU field or periodic group field was greater than 191, or an index of zero was supplied. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41012E PE UPPER INDEX GT 191 OR MU UPPER INDEX GT 191

Explanation	The upper index supplied for an MU field or periodic group field range was greater than 191, or an index of zero was supplied. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41013E INVALID RANGE; FROM NOT LT TO

Explanation	A range was specified in which the lower index was not less than the upper index. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41014E NO MU RANGE ALLOWED FOR PE GROUP FIELD

Explanation	A periodic group field was specified that contains an MU field. This is an illegal usage. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41015E MU IN PE HAS NO SECOND INDEX OR GT 191

Explanation	An index range in parentheses (MU in PE) was invalid. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41016E SECOND INDEX IN RANGE IS INVALID

Explanation	The upper index supplied for an MU field or periodic group field range was greater than 191, or an index of zero was supplied. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41017E INDEX SUPPLIED FOR FIELD NOT MU OR PE

Explanation	An index was specified for a simple field that is neither MU or PE. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41018E 1-N SYNTAX UNSUPPORTED FOR MU WITHIN PE

Explanation	The format buffer syntax *field_name1-N(1-N)* is not supported. It is not possible to display all occurrences of an MU in all occurrences of a PE.
Action	Correct the error and resubmit the request.

DAT41021E COUNT SPECIFIED FOR NON-REPEATING FIELD

Explanation	A count (trailing C) was requested for a field that is neither an MU nor a PE field. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41022E INDEX REQUIRED FOR COUNT OF MULT IN PE

Explanation	When a count is requested for an MU field within a periodic group, the index of the periodic group field must be specified. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41023E ONLY ONE INDEX ALLOWED FOR COUNT

Explanation	A secondary index was specified for a count field. This is an illegal usage. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41024E NO INDEX ALLOWED FOR THIS COUNT

Explanation	An index was specified for a count for a field that is not a multiple value field in a periodic group. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT41031E ALPHA FIELD CANNOT BE CONVERTED TO NUMERIC

Explanation	A format of A was specified for a field that is defined as numeric. This conversion is not allowed. The message indicates the location of the error in the control statement. Execution terminates with a condition code of 12 following analysis of the control cards.
Action	Correct the error and execute the program again.

DAT80000I SAVEFDTS PROCESSING SUMMARY

Explanation	This is the page header message for the SAVEFDTS program.
Action	None required.

DAT80001I *n* FIELDS IN FDT FOR FILE *n*

Explanation	The indicated number of field definitions were found for the Adabas file number shown.
Action	None required.

DAT80002E ADABAS RESPONSE CODE *n* FOR ADABAS FILE *n*

Explanation	The indicated Adabas response code was returned when a call was made to Adabas to read the field definitions for the Adabas file number shown. SAVEFDTS terminates with a completion code of 16.
Action	Correct the error indicated in the Adabas manual and resubmit the job.

DAT80003E ADABAS IS NOT ACTIVE

Explanation	Adabas must be active for SAVEFDTS to be able to read field definitions. Such was not the case. SAVEFDTS terminates with a completion code of 16.
Action	Resubmit the job when Adabas is active.

DAT80004I PROCESSING TERMINATED WITH RETURN CODE *n*

Explanation	SAVEFDTS has completed processing. The return code shown is returned to the operating system.
Action	None required.

DAT80005I PERMANENT I/O ERROR ON SYSUT1

Explanation	A permanent I/O error has occurred when Unicenter CA-PLEU tried to write the field definitions to the SYSUT1 output data set. Execution terminates with a completion code of 16.
Action	Correct the error with the data set and execute the program again.

DAT80010I SAVEFDTS CONTROL CARD INPUT

Explanation	Processing of SAVEFDTS control cards is summarized by the following messages.
Action	None required.

DAT80011I PASSWORD VALUE READ FOR ALL FILES

Explanation	A global PASSWORD parameter card has been read for SAVEFDTS processing. The password is not printed for security reasons.
Action	None required.

DAT80012I PASSWORD VALUE READ FOR FILE *n*

Explanation	A PASSWORD and a FILE parameter card has been read for SAVEFDTS processing. The password is not printed for security reasons.
Action	None required.

DAT80013I INCORRECT KEYWORD FOUND. INPUT CARD IMAGE SKIPPED.

Explanation	An invalid input parameter card has been read. The input card is not printed for security reasons. The placement of the incorrect card in the input stream may be inferred from preceding messages. SAVEFDTS terminates with a response code 16.
Action	Correct and resubmit.

DAT80014I UNCORRECTABLE I/O ERROR ENCOUNTERED READING INPUT FILE

Explanation	A permanent I/O error occurred reading the SAVEFDTS input file. SAVEFDTS terminates with a response code 16.
Action	Verify the JCL and file characteristics of the input file.

DAT80015I SAVEFDTS EXECUTING IN *type* USER MODE UNDER ADABAS VERSION *n*

Explanation	The SAVEFDTS program executed in the indicated user mode against a database of the indicated Adabas version.
Action	None required.

DAT80016I MAXFILES SET TO *n*

Explanation	The SAVEFDTS program unloads the FDTs for files 1-nnnnn. If there are files with valid FDTs beyond the MAXFILES value, these FDTs are NOT unloaded to the SAVEFDTS file.
Action	None required.

DAT80020E MAXFILES GREATER THAN 4095

Explanation	The MAXFILES value supplied exceeds 4095. The Utility terminates with a completion code of 16.
Action	Correct the MAXFILES value to 4095 or less.

DAT80021E PW FILE GREATER THAN MAXFILES

Explanation	The FILE value supplied in the PASSWORD statement exceeds the MAXFILES value. The Utility terminates with a completion code of 16.
Action	Correct the FILE value or increase the MAXFILES value.

DAT85001I FDT USED FOR ADABAS FILE *n* CREATED ON yyyy-mm-dd AT hh:mm:ss

Explanation	The field definition for the indicated Adabas file was read from a sequential file. The information was read from Adabas on the date and at the time shown in the message.
Action	None required.

DAT85002E PERMANENT I/O ERROR READING FDT FOR ADABAS FILE *n* USING DDNAME *ddname*

Explanation	A permanent I/O error has occurred when Unicenter CA-PLEU tried to read the field definition for the indicated Adabas file number from the data set pointed to by the ddname shown. Execution terminates with a completion code of 16.
Action	Verify that the data set provided was created by SAVEFDTS.

DAT85003E FDT READ FOR ADABAS FILE *n* USING DDNAME *ddname* HAS NO ENTRIES

Explanation	The field definition that was read for the indicated Adabas file number from the data set whose ddname is shown has no entries. That is, the Adabas file was not defined at the time that the data set was created by SAVEFDTS. Execution terminates with a completion code of 16 because is impossible to process records for the indicated file.
Action	Verify that the correct Adabas file number was specified on the extract statement. If so, obtain the field definitions from a SAVEFDTS file created at a later date or use the field definitions presently within Adabas.

DAT85004E FDT FOR ADABAS FILE *n* NOT FOUND ON DDNAME *ddname*

Explanation	The field definition for the indicated Adabas file number could not be found on the data set pointed to by the ddname shown. Execution terminates with a completion code of 16 because is impossible to process records for the indicated file.
Action	Verify that the indicated data set was created by SAVEFDTS and that it is complete.

DAT85005E ADABAS RESPONSE CODE *n* WHEN READING THE FDT FOR FILE *n*

Explanation

Unicenter CA-PLEU received the indicated response code from Adabas when it attempted to read the field definitions for the file number shown. The read being attempted was through an LF command presented to Adabas through a direct call. Execution terminates with a completion code of 16 because no records can be processed for the indicated file.

Action

Correct the Adabas problem indicated and resubmit the job.

DAT85006E ADABAS RESPONSE CODE *n* ON THE OPEN CALL

Explanation

Unicenter CA-PLEU received the indicated response code from Adabas on the initial open call. Unicenter CA-PLEU terminates execution with a completion code of 16 because the database cannot be opened.

Action

Verify JCL and control statements and resubmit the job.

DAT97001I EXTRACTS FOR FILE *n*

Explanation

The following messages give information on the extracts requested for the Adabas file with the specified file number.

Action

None required.

DAT97002I DDNAME: *ddname*

Explanation

The name on the JCL DD statement for the extract is indicated in the message.

Action

None required.

DAT97003I BEFORE FORMAT:

Explanation

The following line lists the format buffer specification for the before images for this extract.

Action

None required.

DAT97004I AFTER FORMAT:

Explanation

The following line lists the format buffer specification for the after images for this extract.

Action

None required.

DAT97005I AFTER FORMAT IS IDENTICAL TO BEFORE FORMAT

Explanation

This message occurs when the FORMAT keyword is used. The format of the output records for the before images and the after images are the same.

Action

None required.

DAT97006I LENGTH OF OUTPUT RECORD FOR THE BUFFER IS *n*

Explanation

The record length for the output specified by the preceding format buffer is shown. This is the sum of the lengths of all the fields specified plus the length of the record header.

Action

None required.

DAT97007I ESTIMATED LENGTH OF OUTPUT RECORD IS *n*

Explanation

Unicenter CA-PLEU has estimated the length required to contain the output records that are to be generated for this request. If this length proves insufficient, message DAT99003E is issued. Two things could cause this estimate to be too small:

- Large numbers of occurrences for MU fields or PE groups
- Field lengths larger than the field length specified in the FDT

Action

If the estimate appears that it may be too small, specify a larger size using the RECSIZE parameter on the EXTRACT statement.

DAT97008I ESTIMATED LENGTH OF OUTPUT RECORD FOR UNLOAD OF AFTER/BEFORE RECORD IS *n*

Explanation

For an UNLOAD, ADDFORM, or ADDFORM-LONG request Unicenter CA-PLEU has estimated the maximum length of an output record. If this length proves insufficient, message DAT99003E is issued. Two things could cause this estimate to be too small:

- Large numbers of occurrences for MU fields or PE groups
- Field lengths larger than the field length specified in the FDT

Action

If the estimate appears that it may be too small, specify a larger size using the RECSIZE keyword on the EXTRACT statement.

DAT97009I EXTRACT OF TRANSACTION RECORDS OF *type* TYPE

Explanation	Protection Log records of the type indicated are processed for this request. Available types are C1, C3, C5, or ET.
Action	None required.

DAT97010I READING SIBA RECORDS PRODUCED BY ADABAS VERSION *n*

Explanation	The Adabas version is determined by the value specified in the GLOBALS ADAVER parameter. Unicenter CA-PLEU will process the records it encounters based on this Adabas version.
Action	None required.

DAT98001I *n* BEFORE IMAGE RECORDS WRITTEN ON DDNAME *ddname*

Explanation	The indicated number of records were written to the indicated ddname. These records contain decompressed before images for the Adabas file number being summarized.
Action	None required.

DAT98002I *n* AFTER IMAGE RECORDS WRITTEN ON DDNAME *ddname*

Explanation	The indicated number of records were written to the indicated ddname. All records contain decompressed after images for the Adabas file number being summarized.
Action	None required.

DAT98003I *n* RECORDS WRITTEN ON DDNAME *ddname*

Explanation	The indicated number of decompressed records were written to the indicated ddname.
Action	None required.

DAT98004I *n* BLOCKS WRITTEN ON DDNAME *ddname*

Explanation	The indicated number of blocks of decompressed records were written to the indicated ddname.
Action	None required.

DAT98005I PROCESSING SUMMARY

Explanation	This message identifies the beginning of the processing summary for this execution of the program. Following this message are messages giving various record counts, telling the numbers of records that were selected and discarded for various output files, etc.
Action	None required.

DAT98006I *n* BLOCKS READ FROM THE PROTECTION LOG.

Explanation	The indicated number of blocks were read from the Adabas Protection Log.
Action	None required. The message summarizes the input data provided to the program. This message should be examined to verify that an appropriate number of blocks were read. If this value is smaller than expected, you should verify that no volumes were accidentally omitted from the Protection Log input to the program.

DAT98007I *n* DATA RECORDS READ FROM PROTECTION LOG FOR REQUESTED ADABAS FILES.

Explanation	The indicated number of before and after image records were read from the Adabas Protection Log. This count includes records for all Adabas files that were requested on the extract control cards. Protection log records containing before and after images for Adabas files other than those requested on the extract cards are not included in this count.
Action	None required. This message allows you to verify that the expected number of records were examined. If this number is significantly less than expected, you should verify that all of the intended input was read by the program.

DAT98008I *n* DATA RECORDS WERE DISCARDED DUE TO SELECTION CRITERIA

Explanation	The indicated number of before and after records from the Protection Log were discarded due to the selection criteria specified on the SELECT and EXTRACT control cards. This count includes only before and after image records for the Adabas files that were requested on the EXTRACT cards. Before and after images for other Adabas files are not included in this count.
Action	None required. This message allows you to verify that the desired records were included on the output data sets. If the indicated count is larger than expected, you should examine the selection criteria to verify that no error was made in their specification.

DAT98009I *n* DATA RECORDS WERE PROCESSED FOR AN EXTRACT FILE.

Explanation	The indicated number of Protection Log records were selected to be written to at least one extract output file.
Action	None required. You can verify the effect of the selection criteria by comparing this value with those given by messages DAT98008I and DAT98007I.

DAT98010I *n* RECORDS WERE READ

Explanation	The indicated number of Protection Log records for the Adabas file number being summarized were encountered by the program. Subsequent messages show how many records were discarded due to the selection criteria and how many were decompressed into an output data set.
Action	None required. You can verify that a reasonable number of transaction log records for the indicated Adabas file were read.

DAT98011I *n* RECORDS WERE DISCARDED

Explanation	Of the records indicated in message DAT98010I, this number of records were discarded because they did not satisfy the selection criteria for inclusion in any output data set.
Action	None required. You should compare the values in this message with those in messages DAT98010I and DAT98012I to verify that the selection criteria specified are producing the desired results.

DAT98012I *n* PROTECTION LOG RECORDS WERE PROCESSED INTO EXTRACT FILES

Explanation	The indicated number of Protection Log records for the Adabas file number being summarized were selected for inclusion in at least one output data set.
Action	None required. The indicated value together with the values in messages DAT98010I and DAT98011I can be used to verify that the selection criteria are achieving their desired effect.

DAT98013I FIRST TIME STAMP ENCOUNTERED *yyyy-mm-dd hh:mm:ss*.

Explanation	The time stamp on the first protection block read has the indicated year, month, day, hour, minute, and second.
Action	None required. You should verify that the Protection Log for the proper time period was read.

DAT98014W TIME STAMPS ENCOUNTERED OUT OF SEQUENCE

Explanation	The program compares the time stamps on successive blocks from the Protection Log. If time stamps out of chronological sequence are encountered, this message is issued along with message DAT98015W. This event results in a condition code of four (4) being set at the completion of the program, provided no errors of higher severity are encountered.
Action	None required. You should verify that extra Protection Log records were not accidentally included in the program's input.

DAT98015W FIRST TIME STAMP=*yyyy-mm-dd hh:mm:ss* SECOND STAMP=*yyyy-mm-dd hh:mm:ss*

Explanation	This message gives the dates and times of the two time stamps that were out of chronological sequence. Since Adabas creates its Protection Log with records in chronological sequence, this message occurs only if volumes containing log records are presented out of the sequence in which they were created. This event results in a condition code of 4 being set at the completion of the program, provided no errors of higher severity are encountered.
Action	None required. You should verify that extra Protection Log records were not accidentally included in the program's input.

DAT98016I LAST TIME STAMP ENCOUNTERED *yyyy-mm-dd hh:mm:ss*

Explanation	The time stamp on the last Protection Log record read is presented in the form of year, month, day, hour, minute, and second.
Action	None required. You should verify that records for all desired time periods were provided to the program.

DAT98017I SUMMARY FOR ADABAS FILE *n*

Explanation	This is a header message that precedes the messages which summarize the program's execution for the indicated Adabas file.
Action	None required.

DAT98018I SUMMARY FOR DDNAME *ddname*

Explanation	This is a header message that precedes the message which summarizes the contents of the indicated output data set.
Action	None required.

DAT98019E PERMANENT I/O ERROR READING THE PROTECTION LOG

Explanation	An unrecoverable error has occurred reading the Protection Log. This is probably caused by a hardware malfunction or by damage to the magnetic media that was being read. Processing is terminated with a completion code of 16.
Action	Verify that the data set provided as Protection Log input is in fact a Protection Log. Once this has been verified, attempt to run the job again using a different device, if possible.

DAT98020E PROCESSING TERMINATED

Explanation	This message accompanies message DAT98019E and indicates that the processing of the program is terminated immediately. Processing terminates with a condition code of 16.
Action	See message DAT98019E.

DAT98021E PERMANENT I/O ERROR WRITING DDNAME *ddname*

Explanation	A permanent error occurred while writing the extract file on the device indicated by the ddname. Processing is terminated for this ddname, but processing will continue for other output files. This error results in a return code of eight (8).
Action	This is a probable hardware error. Retry the job using a different output device or volume.

DAT98022E PROCESSING TERMINATED FOR THIS DDNAME.

Explanation	This message accompanies message DAT98021E. See explanation for message DAT98021E.
Action	See message DAT98021E.

DAT98023I *n* BEFORE IMAGE RECORDS WRITTEN FOR ADABAS FILE *n*

Explanation	The indicated number of before images were written to the ddname being summarized for the Adabas file number shown.
Action	None required.

DAT98024I *n* AFTER IMAGE RECORDS WRITTEN FOR ADABAS FILE *n*

Explanation	The indicated number of after images were written to the ddname being summarized for the Adabas file number shown.
Action	None required.

DAT98025W RETURN CODE OF DECIMAL *n* FROM EXIT x. TREATED AS ZERO.

Explanation	A user exit has supplied an invalid return code. A return code of zero has been used instead. This results in the transaction log record being written to the output data set.
Action	Correct the value of the return code being supplied by you exit.

DAT98026I *n* PROTECTION LOG RECORDS DELETED BY USER EXIT x FOR ADABAS FILE *n*

Explanation	The indicated number of Protection Log records for the Adabas file number shown were deleted by you exit whose name is given in the message.
Action	None required.

DAT98027I *n* RECORDS INSERTED BY USER EXIT x FOR ADABAS FILE *n*

Explanation

The indicated number of records were inserted into the data set whose ddname is being summarized. These records were inserted while processing the Adabas file number given in the message and were created by you exit whose name is show.

Action

None required.

DAT98028I *n* RECORDS WERE INSERTED BY USER EXIT x FOR DDNAME *ddname*

Explanation

The indicated number of records were inserted into the output data set whose ddname is shown by you exit whose name is given. These records were created when processing the Adabas file number that is currently being summarized.

Action

None required.

DAT98029I *n* PROTECTION LOG RECORDS DELETED BY USER EXIT x FOR DDNAME *ddname*

Explanation

The indicated number of Protection Log records were deleted by you exit whose name is shown. These records were for the Adabas file number being summarized and would have been written to the ddname shown.

Action

None required.

DAT98030W ONE OR MORE DATA RECORDS FOR ADABAS FILE *n* ARE OLDER THAN THE FDT READ FROM DDNAME *ddname*

Explanation

The time and date stamp on a Protection Log record indicates that it is older than the field definition which was read from the indicated ddname. Message can occur when an FDT is read from sequential file created by SAVEFDTS.

The date that the FDT was read is compared with the Protection Log record date. If the FDT is newer, it might not contain the same FDT as when the Protection Log record was created. In that case, the decompressed data might not be correct. Processing continues. If no errors of higher severity are encountered, the program terminates with a condition code of 4. This message is printed only for the first occurrence of the condition for each Adabas file.

Action

Verify that the correct Field Definition Table was used.

DAT98031W FDT DATE AND TIME *yyyy-mm-dd hh:mm:ss*

Explanation	This is a continuation of message DAT98030W. It tells the date and time that the Field Definition Table was read by program SAVEFDTS. See message DAT98030W.
Action	See message DAT98030W.

DAT98032I SUMMARY FOR *n* RECORDS

Explanation	This is the header message that precedes the message that summarizes the program's execution. Messages giving various record counts, telling the number of records that were selected and discarded for various output files, etc., will follow.
Action	None required.

DAT98033I *n* RECORDS READ FROM PROTECTION LOG

Explanation	The indicated number of records was read from the Adabas Protection Log.
Action	None required. The message summarizes the input data provided to the program. This message should be examined to verify that an appropriate number of records were read. If this value is smaller than expected, you should verify that no volumes were accidentally omitted from the Protection Log input to the program.

DAT94034I *n* RECORDS WERE PROCESSED FOR AN EXTRACT LOG

Explanation	The indicated number of Protection Log records was selected to be written to at least one extract output file.
Action	None required. You can verify the effect of the selection criteria by comparing this value with those given by messages DAT98033I and DAT98035I.

DAT98035I *n* RECORDS WERE DISCARDED DUE TO SELECTION CRITERIA

Explanation	The indicated number of records was discarded due to the selection criteria specified on the SELECT and EXTRACT control cards.
Action	None required. This message allows you to verify that the desired records were included in the output data sets. If the indicated count is larger than expected, you should examine the selection criteria to verify that no error was made in their specifications.

DAT98036I *n* RECORDS WRITTEN ON DDNAME *ddname*

Explanation	The indicated number of decompressed records was written to the indicated ddname.
Action	None required.

DAT99001E ERROR DETECTED IN COMMAND STREAM, PROCESSING ENDED

Explanation	Processing is terminated due to one or more syntax errors in the command statements. Processing is terminated with a condition code of 12.
Action	Resubmit the job after correcting the syntax errors described by previous messages.

DAT99002E ERROR DETECTED IN FORMATS, PROCESSING ENDED

Explanation	One or more errors were detected in the format buffer specifications. These errors are described by previous error messages. Processing is terminated with a condition code of 12.
Action	Resubmit the job after correcting the syntax errors described by previous messages.

DAT99003E OUTPUT BUFFER TOO SMALL FOR x IMAGE FOR ADABAS FILE *n* WRITING DDNAME *ddname*

Explanation	The utility was decompressing a before or after image for the indicated Adabas file using the UNLOAD, ADDFORM or ADDFORM-LONG format. The buffer space provided for building the output record was not sufficient to hold the decompressed record. Because the contents of the data set cannot be completed, processing is terminated for the ddname shown. This message is followed by message DAT99004E. Processing is terminated for this ddname and the utility completes with a condition code of at least eight (8).
Action	Increase the amount of buffer space provided for building this record using the RECSIZE parameter on the correct EXTRACT statement.

DAT99004E PROCESSING TERMINATED FOR THIS DDNAME

Explanation	This message accompanies message DAT99003E. See message DAT99003E.
Action	See message DAT99003E.

DAT99005W DATA LOST THRU TRUNCATION FOR FIELD x IN ADABAS FILE *n* FOR DDNAME *ddname*

Explanation	The indicated field in the Adabas file number shown was decompressed in response to an UNLOAD format request. The field length specified in the Field Definition Table (FDT) was not large enough to hold the value of the field found on the Protection Log. Unicenter CA-PLEU truncates the right-most characters from an alphanumeric field and the left-most bytes from all other fields. Unicenter CA-PLEU continues to process records, but has a completion code of at least 4.
Action	If the data that has been lost must be recovered, the FDT must be altered to give a field length sufficiently large to contain the decompressed data.

DAT99006W x IMAGE FOR RECORD WITH TIME STAMP yyyy-mm-dd hh:mm:ss

Explanation	This message accompanies message DAT99005W and identifies the Protection Log record which required the truncation for the field described by message DAT99005W. See message DAT99005W.
Action	See message DAT99005W.

DAT99007W NO MORE TRUNCATION MESSAGES FOR THIS EXTRACT

Explanation	Messages DAT99005W and DAT99006W are given only 25 times for each extract data set. A count of the number of fields truncated are kept and printed in the processing summary. Messages DAT99005W and DAT99006W are not printed for this extract data set for the remainder of this execution.
Action	None required.

DAT99008W *n* DATA FIELDS TRUNCATED DURING DECOMPRESSION FOR ADABAS FILE *n*

Explanation	This message is part of the summary for the ddname given in message DAT98018I. It tells how many data fields had to be truncated during decompression because the field length provided was not large enough to hold the decompressed data. The completion code for Unicenter CA-PLEU is at least four (4).
Action	Previous messages describes the first 25 fields which were truncated. You should determine whether significant data has been lost. It so, a new format specification or FDT should be built before executing the program again.

DAT99009W *n* DATA FIELDS TRUNCATED DURING DECOMPRESSION FOR DDNAME *ddname*

Explanation	This message is part of the summary for the Adabas file number given in message DAT98017I. See message for DAT99008W for a further explanation.
Action	See message DAT99008W.

DAT99010W DATA FIELD x FOR ADABAS FILE *n* WAS TRUNCATED DURING DECOMPRESSION FOR DDNAME *ddname*

Explanation	The indicated data field for the Adabas file number shown was being decompressed according to the format specification provided by you for this ddname. The value would not fit into the space provided, and therefore had to be truncated. Alphanumeric fields are truncated leaving the left-most bytes intact. All other fields are truncated leaving their right-most bytes intact. Execution continues, but a completion code of at least 4 is returned at the end of the program.
Action	You should determine whether significant data has been lost. If so, the program should be executed again with a revised format specification that provides enough space for decompressing the data.

DAT99011W x IMAGE FOR RECORD WITH TIME STAMP *yyyy-mm-dd hh:mm:ss*

Explanation	This message accompanies message DAT99010W and identifies which Protection Log record encountered the need for truncation.
Action	See message DAT99010W.

DAT99012W NO MORE TRUNCATION MESSAGES FOR THIS EXTRACT

Explanation	Messages DAT99010W and DAT99011W will be printed only 25 times for each extract file. After that point, execution will continue and a count of field truncations will be kept. This count will be printed during the processing summary.
Action	None required.

DAT99013E BLOCK SIZE OF *n* FOR DDNAME *ddname* TOO SMALL FOR MAXIMUM RECORD LENGTH OF *m*

Explanation	Unicenter CA-PLEU attempted to assign the largest possible blocksize for the device allocated for the indicated ddname. This blocksize, which is shown in the message, is not large enough for the largest possible record which could be written to this data set. The output data set cannot be used. Therefore, the entire execution is terminated before any records are read from the Protection Log.
Action	If the maximum record length has been estimated by the program as much larger than necessary, enter a better estimate as the value of RECSIZE on the appropriate EXTRACT statement. If a value of RECSIZE has caused the large record length, either revise the value of RECSIZE or move the output data set to a device that processes a blocksize as large as is required. Tape data sets can have blocksizes as large as 32,760.

DAT99014E PROCESSING TERMINATED DUE TO ERRORS DURING DATASET INITIALIZATION

Explanation	One or more error messages precede this one that describe error conditions which prevent successful execution of the program. Processing is terminated. The completion code has been set according to the error which caused termination.
Action	Correct the error and execute the program again.

DAT99015W ACTUAL DATA LENGTH GREATER THAN FDT LENGTH FOR FIELD *x* IN ADABAS FILE *n*

Explanation	During the decompression of a Protection Log record for an ADDFORM-LONG extract, the indicated field was encountered having a length longer than that specified in the FDT for the indicated Adabas file. Processing continues without error.
Action	None required. However, you might wish to examine the data records to verify the integrity of the data. You might also wish to alter the field length as defined in the FDT so that it conforms with the actual field length in this record.

DAT99016W *n* DATA FIELDS LONGER THAN LENGTH IN FDT FOR ADABAS FILE *n*

Explanation	This is a summarization message that is written at the end of execution of Unicenter CA-PLEU for an ADDFORM-LONG extract. It shows the number of data fields that have actual lengths greater than the corresponding lengths specified in the FDT for the indicated Adabas file. Processing continues.
Action	None required. However, you might want to examine the data records to verify the integrity of the data. You also might want to alter the field length as defined in the FDT so that it conforms with the actual field lengths.

DAT99017W *n* DATA FIELDS LONGER THAN LENGTH IN FDT FOR DDNAME *ddname*

Explanation	This is a summarization message that is written at the end of execution of Unicenter CA-PLEU for an ADDFORM-LONG extract. It shows the number of data fields that have actual lengths greater than the corresponding lengths in the FDT. The summary is for all records written to the indicated ddname. Processing continues.
Action	None required. However, you might want to examine the data records to verify the integrity of the data. You also might want to alter the field length in the FDT so that it conforms with the actual length of the fields encountered.

DAT99018W INVALID RECORD ENCOUNTERED IN PROTECTION LOG

Explanation	A record that was read from what is supposed to be the Protection Log is not in the format of an Adabas Protection Log record. The remainder of the block is skipped after writing message DAT99019W.
Action	Examine the Protection Log record input data set. It contains one or more records that are not valid Protection Log records. Correct the input data and re-execute the program.

DAT99019W SKIPPING REMAINDER OF BLOCK

Explanation	A record that was read from what is supposed to be the Protection Log is not in the format of an Adabas Protection Log record. The remainder of the block is skipped after writing message DAT99019W.
Action	Examine the Protection Log record input data set. It contains one or more records that are not valid Protection Log records. Correct the input data and re-execute the program.

Index

Computer Associates™